平成 15 年度卒業論文

「WEB サービスを利用した顔認識アプリケーションの構築」

学籍番号　200000947
所属　　筑波大学第三学群社会工学類
主専攻　　経営工学専攻
氏名　　飯塚　岳郎
指導教官　佐藤　亮

研究目的

　WEB サービス技術を携帯電話上で利用することを通じて、現在実現可能なユビキタスコンピューティングの手法とその実際を学習する。

研究の特色

　携帯電話上で WEB サービス技術を利用方法について、そのために必要な要素技術について簡潔にまとめた。そして、従来はほとんど不可能であった画像処理のような複雑で計算負荷が大きなサービスを携帯電話に提供することを可能にした。

　また、携帯電話で WEB アプリケーションを構築する際にも、設計手法として UML を全面的に活用できることを示した。

結論

　携帯電話上で WEB サービス技術を利用するとはどういうことか、それがどういう可能性を持つものなのか、またどの程度困難なものであるのか経験とノウハウを得られた。

　その過程で、WINDOWS と携帯電話双方のプログラミング環境と通信技術、WEB サービス技術について、基本的な事柄を理解し実践することが出来た。

平成 15 年度　卒業論文

WEB サービスを利用した顔認識アプリケーションの構築

筑波大学　第三学群　社会工学類　経営工学専攻
学籍番号　200000947
飯塚　岳郎
担当指導教官　佐藤　亮
2004 年 1 月 28 日提出

目次

第 1 章　序論

1.1　携帯電話上で WEB サービスが利用出来ることの意義 …… 3
1.2　研究計画の概要と最終的な成果 …… 4

第 2 章　顔認識アプリケーション作成に必要な要素技術

2.1　WINDOWS プログラミングにおける基本的な概念 …… 6
2.2　携帯電話上のプログラミング環境 …… 11
2.3　WEB サービス技術について …… 13
2.4　顔認識という作業について …… 18

第 3 章　顔認識アプリケーションの分析と設計

3.1　顔認識アプリケーションの想定ビジネスモデル …… 20
3.2　UML を利用したシステムの分析 …… 21
3.3　UML による顔認識アプリケーションの外部設計 …… 22

第 4 章　顔認識アプリケーションの実装

4.1　顔認識アプリケーションに必要な要素技術の習得と利用
4.1.1　画像処理ライブラリ OPENCV とその具体的な実装方法 …… 30
4.1.2　携帯電話上で実際にシステムを構築するにあたっての課題 …… 35
4.1.3　ASPNET と ADONET について …… 40
4.2　UML を利用したアプリケーションの内部設計 …… 42
4.3　顔認識アプリケーションの動作確認 …… 45

第 5 章　結論

謝辞 …… 50
参考文献 …… 51
付録 …… 56
（付録 1）FaceLibrary2（顔認識ライブラリ）
（付録 2）Webservice1（WEB サービス本体）
（付録 3）KSOAP シリーズ（携帯電話上のクライアント）

第１章　序論

1.1　携帯電話上で WEB サービスが利用出来ることの意義

現在、インターネットの利用が普及し、ビジネスの形態を変化させるだけでなく社会の枠組みさえ変えようとしている。しかし、現在の所、インターネットを利用するということは、携帯電話にせよパソコンにせよ自分の手でキーボードとマウスを使って文字情報をやりとりすることが基本である。

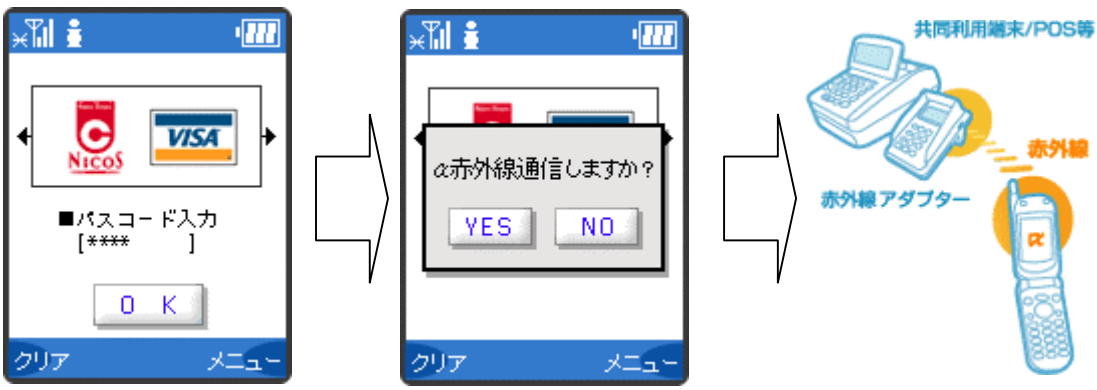
しかし、今日、何らかのプログラミング環境を利用出来る携帯電話は一般化した。これは十分信頼できる通信能力と初期のパソコンに匹敵するだけの処理能力を持つ情報機器を誰もが持ち歩いていることを意味する。

また、従来は全く異なった環境のプログラム同士の対等な情報交換は極めて困難で DNS のようなごく一部でしか実用化されていなかったが、近年、WEB サービス技術が登場し、比較的容易にプログラミングできるようになった。この技術により、コンピュータに自律的な情報処理能力を与えるエージェントシステムが現実に手の届くものとなった。

そして、何より、ハードウェア技術の進歩で、携帯電話にデジタルカメラ、赤外線通信機能、GPS などの通話とメール以外の様々な機能を付加することが可能になった。

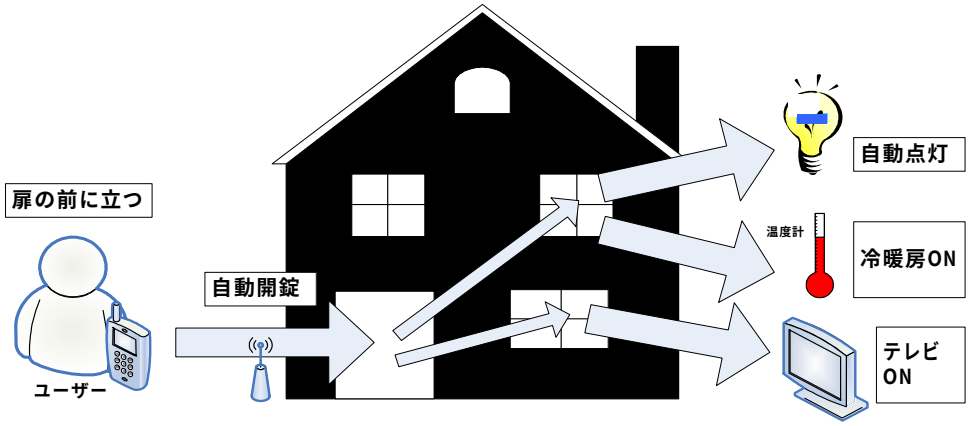
現在でも携帯電話による WEB アプリケーションは赤外線通信によるカード決済は大きな可能性を秘めている（図 1-1 参照）。しかし、以上に述べた 3 技術を組み合わせれば携帯電話上の自律的なエージェントシステム、それこそ自分がある場所へ移動するだけで携帯電話が反応して勝手にサービスを受けられるようなエージェントシステムを構築可能になる。この技術は現在のところ実験段階であるが、今後、ユビキタスコンピューティングの第一歩を築くことになると私は確信している。

（図 1-1）Nicos VISA っぴ（携帯電話によるカード決済の一例）（文献[8-2]より引用）



そこで、私は、このように巨大な可能性を秘めた携帯電話上の WEB サービス技術、及びその周辺に関わる様々な技術を習得することを目的として研究を行うこととした。そのため、応用性の高く、かつ WEB サービス技術なしには通信の難しい画像データを利用する顔認識技術を携帯電話上で提供するシステムを構築することにした。

（図 1-2）携帯電話によるユビキタスアプリケーション（想像図）

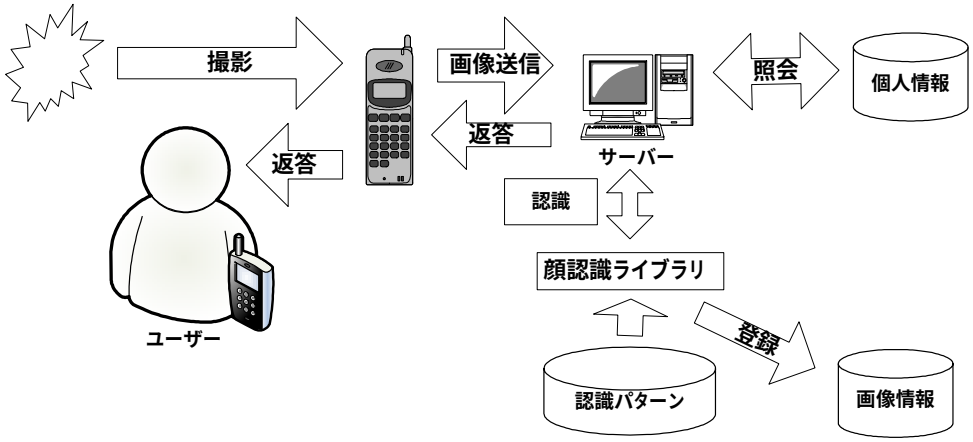


1.2　研究計画の概要と最終的な成果

今回、携帯電話で機能する WEB アプリケーションの作成の一例として、携帯電話で機能する顔認識アプリケーションを作成することにした。

携帯電話で人の顔を撮影すると携帯電話上のプログラムが指定されたサーバに送信し、サーバは予め学習済みのパターン情報を利用して顔認識を行い、結果を送り返すというプログラムである。詳細は以下の図 1-3 に示した。

（図 1-3）顔認識アプリケーションの完成予想イメージ（純粋な顔認識機能のみ）



最終的に、顔認識アプリケーションのうち、図 1-4 の色で囲った部分は完成した。具体的には以下の 3 項目は完成した。

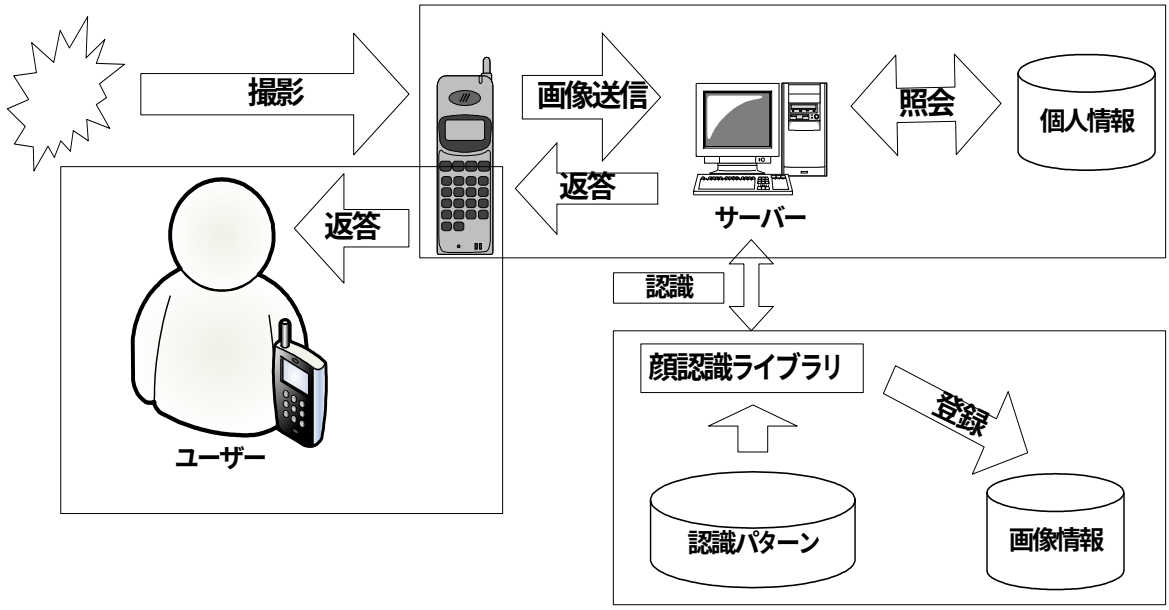
（1） 携帯電話とサーバ間の WEB サービス運用
（2） 顔認識ライブラリ本体
（3） 携帯電話上の GUI

しかし、それ以外の以下に挙げる 2 項目は完成しなかった。

（1） 顔認識ライブラリと WEB サーバの接続
（2） 撮影機能（作成したがエミュレータでは検証不可能）

今回、（1）の問題が解決不能に陥ったため、実機でのテストは無意味と判断して行わなかった。そのため、（2）の問題は検証できなかった。また、同様の理由でプログラムのサイズ制限にも従っていないので完成した部分も実機で作動する保証はない。

（図 1-4）顔認識アプリケーションのうち実際に実装された部分（純粋な顔認識機能のみ）



第 2 章　顔認識アプリケーション作成に必要な要素技術

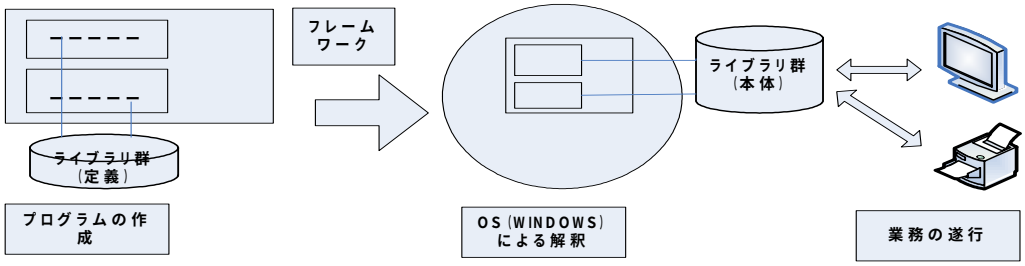
2.1　　WINDOWS プログラミングにおける基本的な概念

2.1.1　MFC フレームワークについて

MFC（Microsoft　Foundation　Class）とは、WINDOWS 上のプログラムを作成するための Microsoft 社提供のフレームワークとクラスライブラリ群のことである。

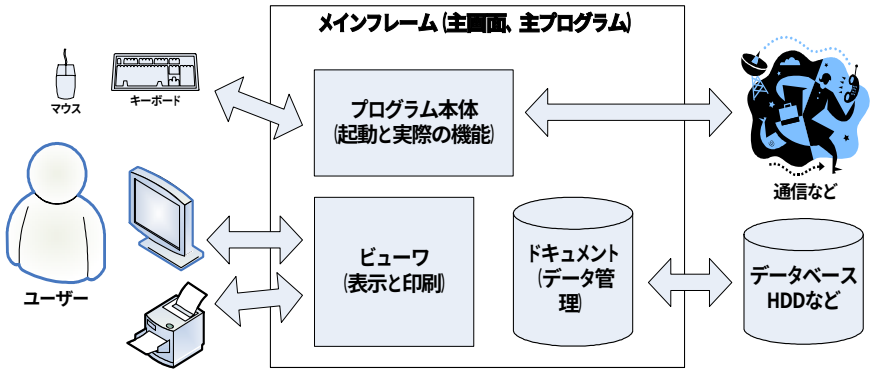
フレームワークというのは、論文における「序論」「本論」「結論」と同様、プログラマーとコンピュータ側が共有するプログラム上の枠組みのことで、その枠組みにそって OS がプログラムを解釈、実行する。クラスライブラリというのはオブジェクト指向におけるクラスの形式にまとめられたプログラム部品のことで、OS が提供する様々な業務（描画や文字の出力など）を提供するものである。

（図 2-1）フレームワークとクラスライブラリの関係



WINDOWS はメッセージ起動の OS であるため、単純なプログラムを作成するにも決まりきった命令を何百行と書かなければいけなかった。これが面倒なので、決まりきった作業をクラスライブラリ化し併せて WINDOWS アプリケーションを標準化したのが MFC である。 NET フレームワークはそのネットワーク版と考えることも出来る。

（図 2-2）MFC フレームワークの概略



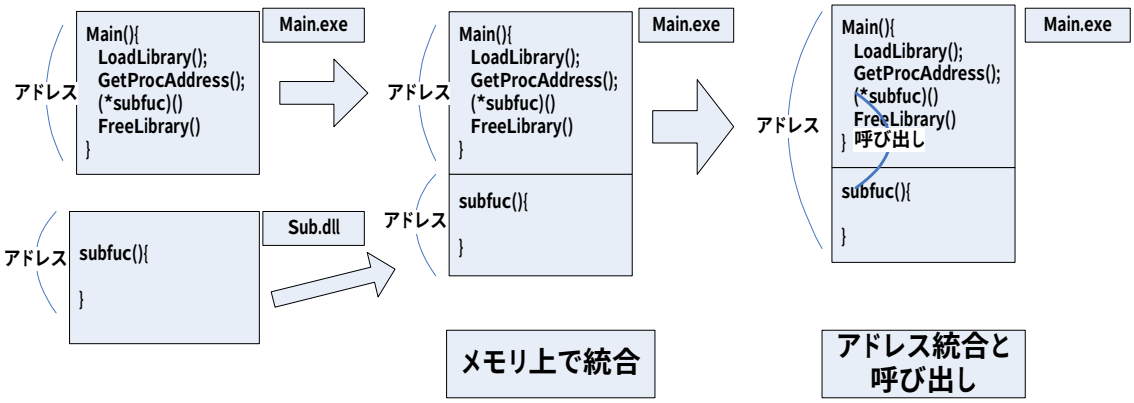
2.1.2　COM（Component Object Model）について

Microsoft 社が提唱する、部品化されたプログラムを作成・利用するための基盤となる技術仕様。特定の機能のみを持つプログラム部品を組み合わせることでアプリケーションソフトの開発が容易になる。（文献[6-7]）

WINDOWS では規格が定められており、実行ファイル（EXE ファイル）とプログラム部品（DLL ファイル）を分離出来るのは良く知られている。しかし DLL ファイルの規格は、当初、関数（プログラム部品）を呼び出すのに必要なメモリ上でのアドレスの統合など、分離したプログラム部品を呼び出すことを保障するものに過ぎなかった。従って、プログラム部品の再利用や他のプログラムからの呼び出しは一切考慮されていなかった。

そのため、プログラム部品の再利用やインターネットなどサーバを介したサービス提供は事実上不可能であった。プログラミングの合理化やインターネットの利用という観点から見て、これは大きな問題であった。

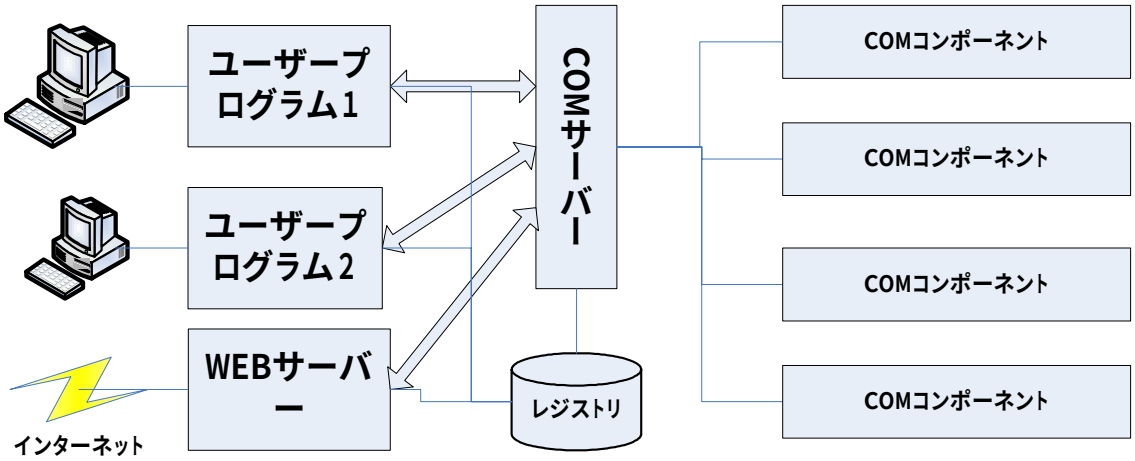
(図 2-3)当初の規格による DLL 呼び出しの概念図（文献[5-4]上の図を改変）



そこで、Microsoft 社はプログラム部品のデータの転送と呼出に関する規格を策定し、合わせて WINDOWS 上に専用のサーバープログラムを設置、登録管理を一任することにした。これが COM 規格であり、この仕様に基づく DLL を COM コンポーネントと呼ぶ。

発想法としては RPC に類似した発想で、この規格に合わせて Microsoft 社は WINDOWS 自体を COM コンポーネントで構成するように改めた。WINDOWS でレジストリと呼ばれるものの大半は、OS の主要機能を果たす COM コンポーネントの登録に携わっている。COM コンポーネントの呼び出しも呼び出し側がレジストリを参照しその内容を COM サーバに伝える形で実現される。

（図 2-4）COM 規格と COM 規格による DLL 呼び出し



このCOMコンポーネントは開発に使った言語やコンポーネントのある場所などによらず、どの言語からでも、同じコンピュータ上ならどこからでも利用することができる。WORD上のデータをクリックするとプログラムが起動するOLEや、インターネットサーバーであるASP（Active Server Page）はこの手法を利用して構築された。その延長として、皆さんも良くご存知と思われるActiveXコントロール（インターネット上で流通、利用可能なCOMコンポーネント、要するにプラグインの類）が開発された。

とはいえ、図2-4を見れば分かるようにCOM規格は結構複雑な枠組みで、それらの枠組みに対応するためCOM規格を遵守したCOMコンポーネントを作成するためには10を超えるクラスや関数の呼び出しが必要で、その作業は容易ではない。

そこで、Microsoft社はATL（Active Template Library）と呼ばれる、Visual C++を使ってActiveXコントロールを作成するための作業をまとめた雛型（テンプレート）とその雛型専用のライブラリ群を提供している。Microsoft .NETが登場する前まではゲームの描画機能にせよDBへのアクセスにせよ、WINDOWSで必要とされる主なDLLファイル（部品プログラム）はこの形式で作成されてきた、

2.1.3 Microsoft .NET フレームワークについて

Microsoft 社が 2000 年 7 月に発表した、ネットワークベースのアプリケーション動作環境を提供するシステム基盤である。（文献[6-7]参考）

COM と同様、登録管理を WINDOWS 上のサーバが行うプログラム部品（DLL ファイル）の規格ある。この NET フレームワークによるプログラム部品を NET アセンブリと呼ぶ。ただし、実行ファイルの記述に中間コードを用い、プログラムの実行も OS 本体ではなく仮想マシン（サーバ）で行う、従来の COM や MFC とは全く次元の異なる規格である。そのため、COM よりも DLL ファイルのプログラム部品化が徹底された。

中間コードの利用という部分でも分かるように JAVA に類似したシステム基盤で、実際仮想マシン（CLR）によって実行される。Microsoft 社版 JAVA と考えて差し支えない。このような中間コードの利用には以下のような大きな利点がある。

（1） 携帯電話から高性能サーバまでハードウェアの差異に関係なく、この基盤に対応したプログラムなら同様のサービスを提供できるようになる。

（2） 中間コードを利用しているため内部にプログラムに関する情報を埋め込むことが可能であり、この情報を元にコンピュータ側が実行を管理するので、コンピュータの安全性やプログラムのバージョン管理が容易になる。

利用言語は VB、C#、VC++（マネージ C++）、その他にも言語（DELPHI や COBOL など）が存在し、それぞれのコンパイラが中間コードを作成する。

この点が JAVA と違い、Microsoft .NET は利用言語が複数存在するのでその長所を組み合わせることが可能である。Microsoft .NET の規格が厳しいので、どの言語を用いても同様の機能を持つプログラムを作成できるためである。

このように中間コードの利用は WEB アプリケーション作成に決定的に有利な性質であるため、ASPNET（WINDOWS 版拡張 CGI）も ADONET（データベースアクセス用コンポーネント）も当然 NET 基盤で構築されている。

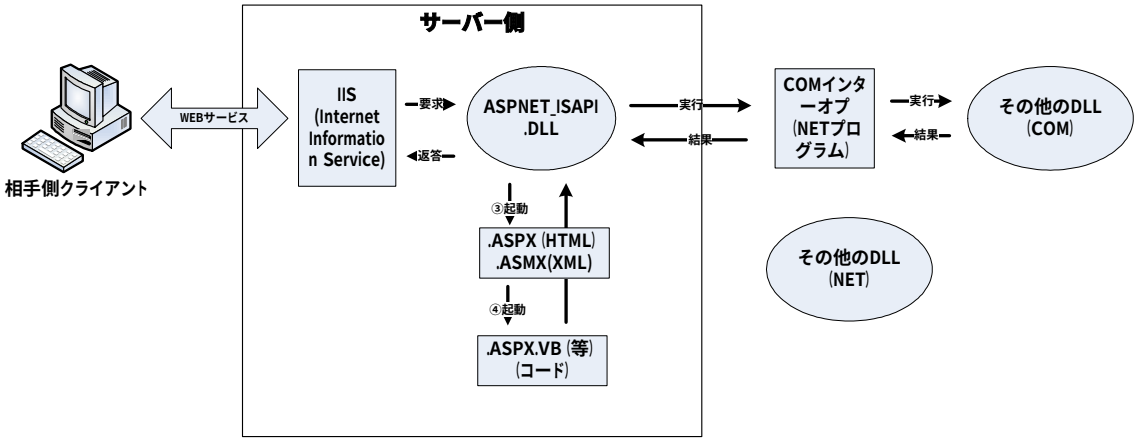
従って、従来の COM 規格で作成したプログラム部品も NET フレームワークで動かせなければ、最新の WEB アプリケーション上では利用できない。しかし、NET 規格による実行ファイルと従来の DLL ファイルとは記述コードさえ異なる別世界の存在である。

そのため、COM 規格による DLL ファイルを NET 基盤で運用する相互運用性と呼ばれる機能が存在し、2 つの手法を選ぶことが出来る。

（1）COM コンポーネントを直接利用できる

（2）COM コンポーネントを NET アセンブリ（NET 形式のプログラム部品）化する

（図 2-5）COM コンポーネントの直接利用の概念図



（図 2-6）COM コンポーネントの NET アセンブリ（NET 形式のプログラム部品）化

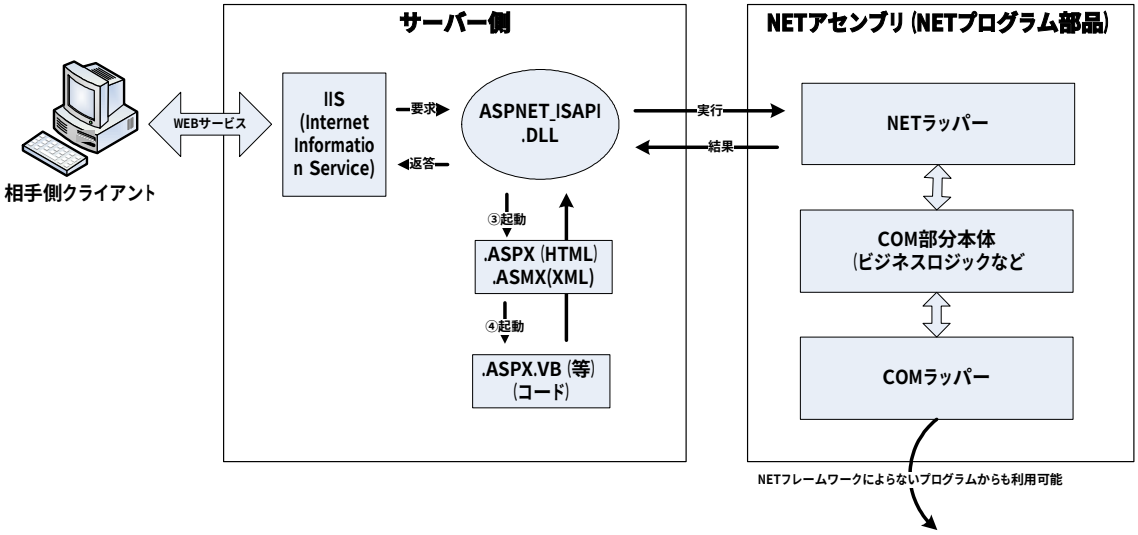


図 2-5 を見れば分かるように、（1）の場合の方がプログラミングは容易である。しかし、COM インターオプと呼ばれる中間プログラムを経由して動作するため性能が低下する。また、ASPNET は必ずしも COM を全面的に受け入れるわけではないという問題も存在する。この様な利害得失をみてプログラマーは既存のプログラムを NET 対応させる必要がある。

2.2　携帯電話上のプログラミング環境

携帯電話上でのプログラミングは、主に、JAVA（JAVA　ME 仕様）を基本とする、携帯電話会社によって独自に拡張された実行環境に対して行われる。プログラマーはこの携帯電話会社の提供した仕様とライブラリに沿ってプログラムを作成する。図 2-7 に NTTDOCOMO の携帯電話の実行環境の概略を示した。

（図 2-7）NTTDOCOMO の携帯電話環境概念図（文献[1-1]より引用）

ただし、携帯電話はバッテリーの消耗を避けるため極めて少ない消費電力で機能する必要がある。そのため電力を多量に必要とする高性能な CPU や大容量のメモリを装備することは困難である。そのため、プログラムの実行速度、利用できる記憶容量は極めて小さい。

（表 2-1）携帯電話のコンピュータとしての性能について（文献[1-1]より引用）

携帯電話の性能	Doja3.0	PC98VM21
CPU周波数	33kHZ	10KHZ未満
主記憶	最低500KB	640KB
実行ファイル　サイズ	30KB	1.4MB（フロッピー）
外部記憶	200KB	1.4MB（フロッピー）
1回の通信制限（送信	10KB	無制限
1回の通信制限（受信	20KB	無制限
通信速度	144Kbps	9.6Kbps

（注）一部の会社はJAVAよりスマートカードへアクセス許可

スペックを見る限り、現在（2004 年度）の携帯電話は 25 年ほど前のパソコンと性能は同等である。ゲーム機でいえばファミコンと同等と考えて構わない。そのような環境で実用的なプログラムを作成するのはそれなりの困難はあるものの不可能ではない。

だが、25 年ほど前のパソコンとは違い、携帯電話は常時携帯するものであり、しかも GPS や赤外線通信、デジタルカメラのようなセンサーも装備している。

（表 3-2）携帯電話に装置された追加機能（2003 年 10 月現在）(文献[7-2]より引用)

	NTT (DOJA)	AU (EZJAVA)
GPS	×	◎
赤外線通信	◎	×
デジタルカメラ	◎	△
bluetooth	◎	×
赤外線リモコン	◎	×

◎はJAVA対応、○は一部対応、△は装置のみ、×は装置

単純に、フォームを利用して文字を入出力する程度の業務なら携帯電話用の WEB アプリケーションを作成し、携帯電話の WEB プラウザ機能からアクセスしたほうが賢明である。しかし、このようなセンサーを制御し利用するためには携帯電話上でプログラミングするしかない。これが携帯電話プログラミングの意義である。

ただし、携帯電話上のセンサーからデータを得られても多くの場合、サーバを呼び出さなければ無意味である。携帯電話上でのキー入力はご存知のように極めて困難なので、人間の手を介さず携帯電話上のプログラムから直接サーバを呼び出す必要がある。それには WEB サービス技術が最適で、それゆえ携帯電話上で WEB サービス技術を利用することに意義がある。

ただし、そのためには WEB サービス技術を用いた通信機能、すなわち SOAP クライアント機能をプログラムに付加なければいけない。

現在では携帯電話上で待ち受け状態からプログラムを起動する機能は備えられ、単にプログラムを実行するだけでなく一種のサーバを携帯電話上に組み立てることが可能である。これと WEB サービス技術を結びつければ一種の自律的なエージェントシステムを構築することさえ可能である。

2.3　WEB サービス技術について

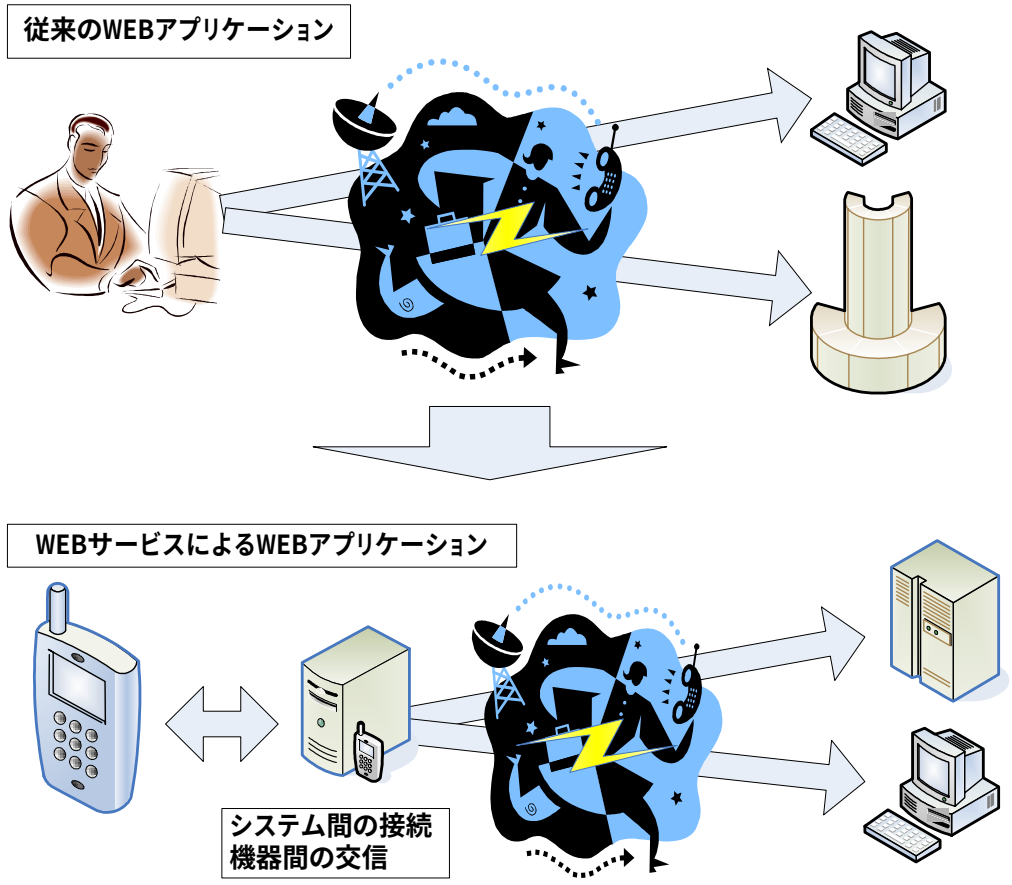
この節では WEB サービスを構成する様々な規格について説明する。なお、この説明を行うにあたり、文献{6-5}を全面的に参考にした。

2.3.1　WEB サービスとは何か

Web サービスとは、WEB 技術などインターネットの通信技術を使ってネットワーク上に分散したアプリケーションを連携させる技術のことである。あるいは、その技術によって連携させられるアプリケーションそのもののことを呼ぶこともある。（文献[2-3]より引用）

この技術は RPC（遠隔メソッド起動）技術の延長で、ある WEB アプリケーションが別のシステム上の WEB アプリケーションを自動的に呼び出すことを可能にする。このことによって、従来の WEB アプリケーションは人間とシステムの間のサービスを提供したのに対し、WEB サービスによる WEB アプリケーションはシステムとシステム同士を接続するサービスを提供する。システムの連携を以前より遥かに容易にする可能性を持つ。

（図 2-8）従来の WEB アプリケーションと WEB サービスによる WEB アプリケーション



Web サービスは、WEB 技術のような通信技術と RPC の延長と言うべきシステム連携技術の 2 つの側面を持つ。従って、広い範囲に及ぶ複合的な技術で構成されている。しかも、発展中の技術で同様の目的を持った複数の競合する規格が存在する分野も多い。

そこで、以降の説明では、Web サービスの中核をなす技術で、しかも仕様が比較的定まっている SOAP、WSDL、UDDI とこの 3 技術を支える基盤技術である XML についてのみ紹介したい。

2.3.2 XML について

XML（eXtended Markingup Language）という規格は、HTML 規格（WEB ページの規格）に、文書の構造化規格 SGML（Standard Generalized Markingup Language）の概念を導入して拡張した規格である。論理や意味による構造化を行う言語なので、ユーザーの需要によって自由に規格を拡張することが可能である。（文献[2-10]より引用）

1980 年代初頭に考案された HTML 規格は、人間が見るための文書の規格として定められたもので、ユーザーによって異なるタグが導入されることを防ぎ HTML 文書の共通性を維持するため、HTML 規格の拡張性やユーザーによって異なる可能性の高い文書の論理構造の定義は意図的に省かれていた。

しかし、機械に HTML 文書を解釈させ業務を実行させようと考えた場合、文書に意味や論理による構造を付与するのは必要不可欠である。筑波大学生は WEB ページに左側に示した番号が表示されれば学籍番号であることを理解できるが、機械は XML を利用しない限り学籍番号であることを理解するのは不可能である。

例）HTML の場合
<H1>200000947<H1>

例）XML の場合
<学籍番号>990964<学籍番号>

今日、HTML 規格による通信は世界標準となった。この通信規格を利用して機械同士を接続する意義は大きいが、HTML 文書を人間が閲覧するのでなく機械が目的を持って処理するのであれば文書の論理構造を記述することが必須である。そのため、HTML 文書にあえて論理構造の定義と拡張性が導入された。それが XML 規格である。

それゆえ、XML 規格は特定の目的を持った機械同士を接続するための規格であり、特定の業界で定められたデータ交換規格の母体として扱われることが多い。
（文献[2-10]より引用）

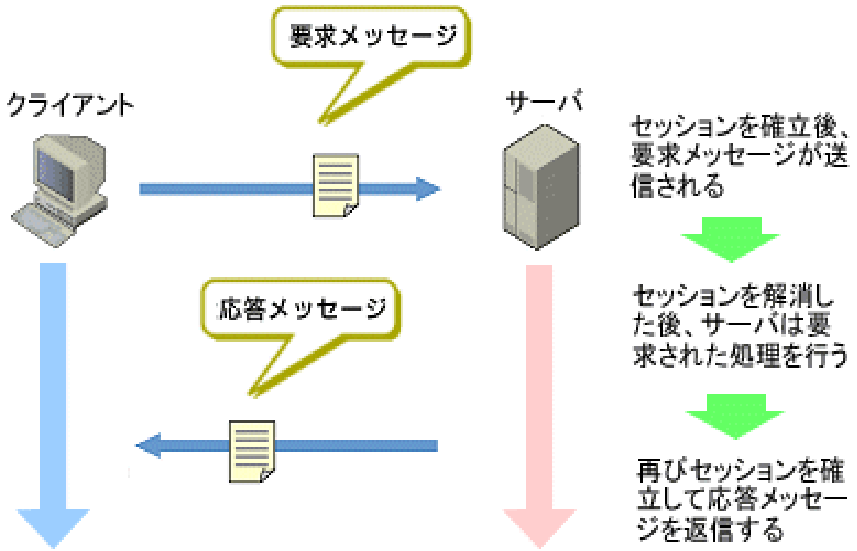
2.3.3　SOAP について

SOAP（Single Object Access Protocol）とは WEB サービス技術の根幹となる規格で、Web サービスで使用されるメッセージのデータフォーマットや、メッセージの処理ルールを定めた通信規約のことである。（文献[2-3]より引用）

WEB サービスは一種の RPC（遠隔メソッド起動）技術であり、あたかも自分の実行環境にあるのと同じように相手のサーバ上の WEB アプリケーションを起動することを可能にする。そのためには以下で示す段階を踏む必要がある。
（１）こちら側のクライアントが起動命令を引き受ける
（２）相手の WEB サーバにメッセージを発信して WEB アプリケーションの実行を依頼
（３）相手の WEB サーバに結果を送り返してもらう
（４）こちら側クライアントで送り返されたメッセージを要求された形式に変換

メッセージそのものは WEB 規格（HTML）やメール規格（SMTP）など前述したような標準的な通信手法で送受信できるが、相手のサーバとの交信を規格化しなければ仕方がない。そのための規格が SOAP（Single Object Access Protocol）である。

（図 2-9）WEB サービスの最も基本的な動作（文献[2-3]より引用）



SOAP メッセージは前述した XML 規格に基づく、UNICODE で記述されたテキストデータである。内容としてはシステムとシステムを結ぶ郵便であり、SOAP 規格は郵便でいうところの封筒の仕様を定めている。なお、データの送受信は WEB 規格やメール規格で対応するので、URL は SOAP 規格では取り扱わない。

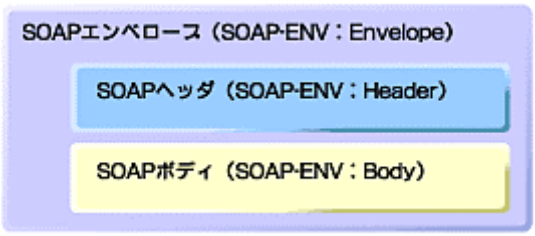
SOAP メッセージの構成

SOAP エンベロープ：郵便で言うところの封筒に相当

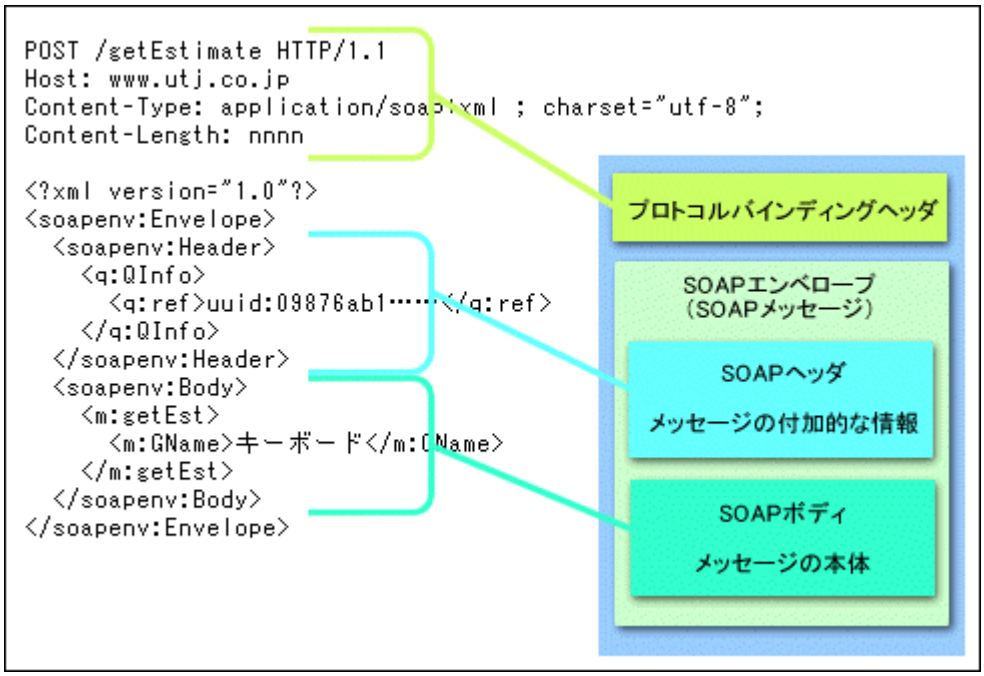
SOAP ヘッダ：ユーザー情報や処理方法など SOAP メッセージのオプションを記載
標準の手続きを利用するなら省略可能

SOAP ボディ：依頼の本文。実行してほしいプログラムと渡すべきデータを記述

（図 2-10）SOAP メッセージの概念図（文献[2-3]より引用）



（図 2-11）実際の SOAP メッセージ（実行依頼側）（文献[2-3]より引用）



2.3.4　WSDL と UDDI について

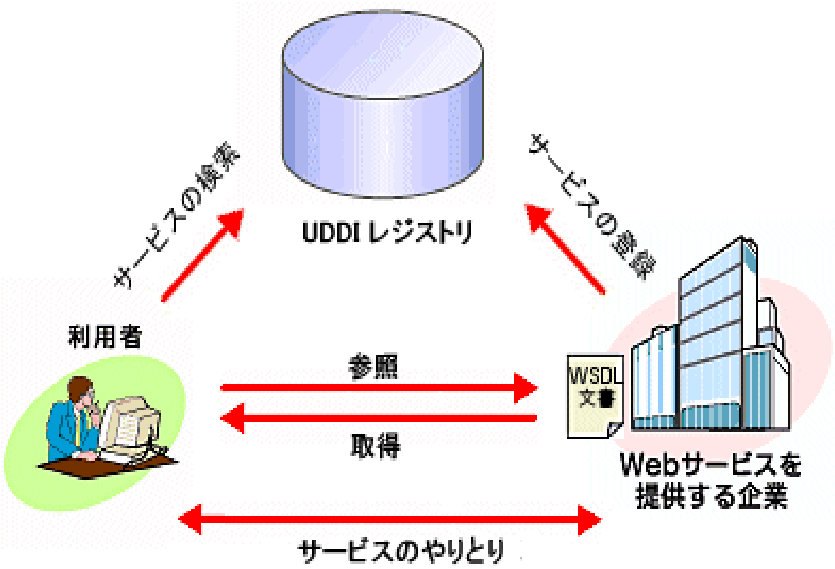
SOAP 規格によって、WEB サーバを通じて相手の WEB アプリケーションを自由に呼び出せるようになった。しかし、単に WEB アプリケーションを自由に呼び出せるというだけでは単純に 2 点間を結ぶ RPC 技術の延長に過ぎず、インターネットを利用しているにも関わらずその性質を生かすことが出来ない。

理由は以下の通りである。
（1） アプリケーションの所在が分からない
（2） アプリケーションの所在が分かっても呼び出し方法が分からない。
（正確に呼び出さない限りアプリケーションは利用不可能である）。

この問題を解決するには自分の作成した WEB アプリケーションを特定のサーバに登録し、定められた規格で利用方法を外部に公開する必要がある。**そのためのサーバに相当する**のが UDDI（Universal Description, Discovery and Integration）であり、利用方法を外部へ公開する規格に相当するのが WSDL（Web Services Description Language）である。ユーザーはこの WSDL 文書を元に SOAP クライアントを設定する。

なお、WSDL 文書、UDDI レジストリ（UDDI そのものはサービスなので、登録文書）とも XML に基づいた構造化文書である。

（図 2-12）WEB サービスの利用法と WSDL,UDDI の関係（文献[2-3]より引用）

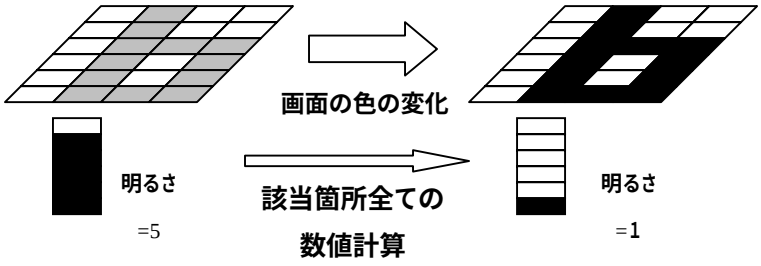


2.4　顔認識という作業について

2.4.1　顔認識ということの概念

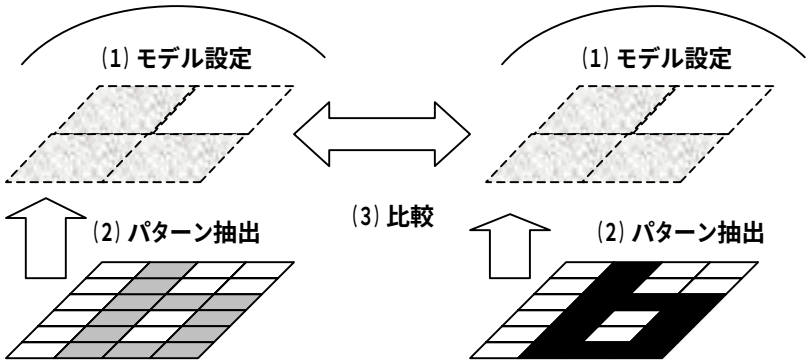
顔認識というのは典型的な画像処理である。 画像処理というのは何であるのかというと、要するに画像を構成するメモリ上にある膨大な数表の操作である。この点に関してはゲーム目的の３D 動画の処理でも顔認識もやっていることは同じである。

（図 2-13）画像処理ということの概念



画像認識とは何か、要するに画像をある種のパターンとみなすモデルを立てパターンを抽出し、SVM や隠れマルコフモデルなどしかるべきパターン認識理論で処理することである。前述したように、コンピュータ上では画像は巨大な数表なので、そのような数表に対するパターン認識である以上３D 動画の処理と同様膨大な計算量を必要とする。

（図 2-14）画像認識の概念と行われる作業



顔認識も画像認識の 1 種類で、20 年近く前から多くの研究所や企業で研究が行われてきたが、郵便番号などの数字とは違って顔ははるかに複雑なパターンでしかも個人個人によって違うためパターンが無数に存在する。これは非常に膨大な計算量を必要とし、実用化が困難であった。それが近年のパターン認識理論の進歩とコンピュータの性能向上の結果、ようやく実用化の域に達し、数年前から製品が市場に出るようになった

2.4.2 顔認識ライブラリの性能と評価

このような顔認識プログラムの性能評価はどう行うか。それは、指定された機材と条件のもと大人数にたいして実験を行い、その結果を**他人許容率**（第 1 種誤り）と**本人拒否率**（第 2 種誤り）という典型的な統計的数値にまとめることで行う。

大手電機各社の顔認識アプリケーション（多くの製品はハードウェア込み）は、スペック表（表 2-3）を見れば分かるとおり、本人認識率（つまり**本人拒否率の逆**）はのきなみ 99%以上であるとのことである。しかし、各社の実験条件はどういうものなのか皆目不明なので、性能評価に疑問が持たれている（文献[7-3]、文献[7-5]の調査結果）

というのは室内の整った環境ではその数値を得られたとしても、現場、特に屋外では性能が大きく低下することが予想され、現にそれが原因で米国の警察が採用した Visionics 社の監視システムが事実上運用停止に追い込まれたという前例がある（文献[7-5]を参考）。

（表 2-3）顔認識技術のスペック表（文献[7-3]より作成）

	開発元	認証精度	認識速度	SDK価格
FaceReco	オムロン	不明	不明	不明
NeoFace	NEC	99%以上	1ミリ秒/1登録者	250万円
FaceIt	Visionics(米)	99%以上	200ミリ秒/1登録者	150万円
FaceX	DDS	多少難	1秒/最終認証	数万円？
Facepass	東芝	99%以上	1秒/最終認証	不明

いずれにせよ、パスワードや鍵の代用に出来るかどうかは疑問である。指紋や虹彩の場合、据付式装置の利用がようやく信頼される認証手段とみなされるに過ぎない。これでも、それだけの誤差がある。顔認識をそれも携帯電話の環境で本格的な認証に用いるのは、今のところ時期早々であると言わざるを得ない。しかし、そこまでの精度を必要としない何かの業務には利用できるのではないかと私は考えた。

（表 2-4）他の技術のスペック表（文献[7-3]、文献[7-4]より作成）

	携帯電話指紋認識	据付式指紋認識	据付式画像認識
本人拒否率	1%	0.004%	1%以下？
他人許容率	0.01%	0.0002%	不明

（据付式装置は富士通のパソコン用 USB センサー）

第 3 章　顔認識アプリケーションの分析と設計

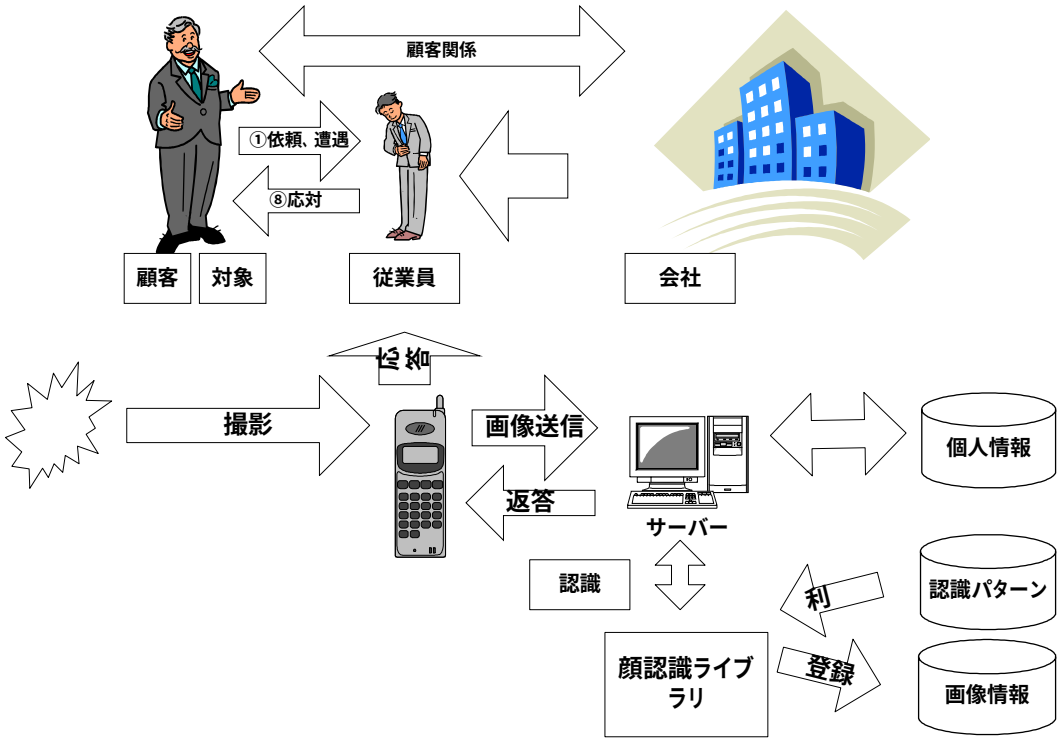
3.1　顔認識アプリケーションの想定ビジネスモデル

現在、先進国では需要が飽和状態になり、需要に対して生産が大幅な過剰となっている。それゆえ、今までのような顧客に大量生産したモノを「売りつける」ビジネスモデルが通用しなくなり、顧客にサービスを提案して「買い上げてもらう」ビジネスモデルを採用しない限り利益を増大させることは困難である。

そこで、顧客、または潜在顧客の情報を収集し、コンピュータにかけて分析し、それから対象を区分してプロモーションをかける CRM（Customer　Relationship　Management）という技術が登場した。

しかし、いくら顧客情報を集めて分析したとしても、目の前にいる顧客が何者なのか識別出来なければ意味がない。例えば旅行代理店ならチケットを見れば顧客が何者か一目瞭然で識別できるものの、例えばレストランの来客は人間の記憶以外に一目で識別する手段は考えにくい。そこで、私は、主に顧客の識別の補助手段として、顔認識アプリケーションを採用することを提案した。以降、このアプリケーションの詳細を設計する。

(図 3-1)想定しているビジネスモデルとシステム概念図



3.2　UML を利用したシステムの分析

3.2.1　UML（Unified　Modeling　Language）とは何者か

オブジェクト指向によるプログラミングに対応する設計手法の標準で、1993 年にオブジェクト技術の国際会議 OOPSLA に初提案され、1997 年にオブジェクト指向技術のための標準化団体 OMG（Object Management Group）に承認された。この設計手法はプログラムの内部設計だけでなく、ビジネスモデルを定義し検討する手段を提供し、それを元に外部設計を行うことが可能な画期的な設計手法である。

1980 年代前半に、オブジェクト指向の概念がプログラミングに導入されシステム開発の標準として確立した。これに対応する形で 1980 年代後半から様々なオブジェクト指向による設計手法が提案されたが、採用する設計手法によって同じ概念の表記が異なるという深刻な問題が生じた。そのため、設計手法の標準化の必要があったので確立された。

今回私がアプリケーション開発に利用する WINDOWS、携帯電話とも、プログラミングは全てオブジェクト指向によるプログラミングを基礎に行われるため、システムの設計手法に UML を採用した。

3.2.2　UML 分析で行うべき作業

UML は与えられたビジネスモデルを表記して検討し、検討してまとまったビジネスモデルを元にプログラムの外部設計（ユーザー側に示す仕様の設計）と内部設計（プログラム本体の設計）を行うための設計手法である。そのため、外部設計と内部設計に当たる作業を遂行しなければいけない。

外部分析に相当する作業はビジネス分析と要求分析とシステム分析の３つに分かれ、それぞれ順に実行する。この 3 つの分析の目的、及び作業内容は表 3-1 に示した。最終的にクラス図とシーケンス図を作成し、実際のシステム開発ではシステム開発に対する顧客の最終許可を得ることが目標となる。

ただし、今回はビジネスモデルの検討は完了したとみなし、図 3-1 を**ビジネスモデルを記述した図面として扱い、要求分析とシステム分析のみ実施するものとする。**

（表 3-1）UML における分析（文献[6-2]より引用）

	目的	作成する図面
ビジネス分析	ビジネスモデルの確定	ビジネスモデルを記述した図面
要求分析	設計対象の切り出し	ユースケース図、ユースケース記述
システム分析	外部設計の確定	クラス図（ビジネス用）、シーケンス図

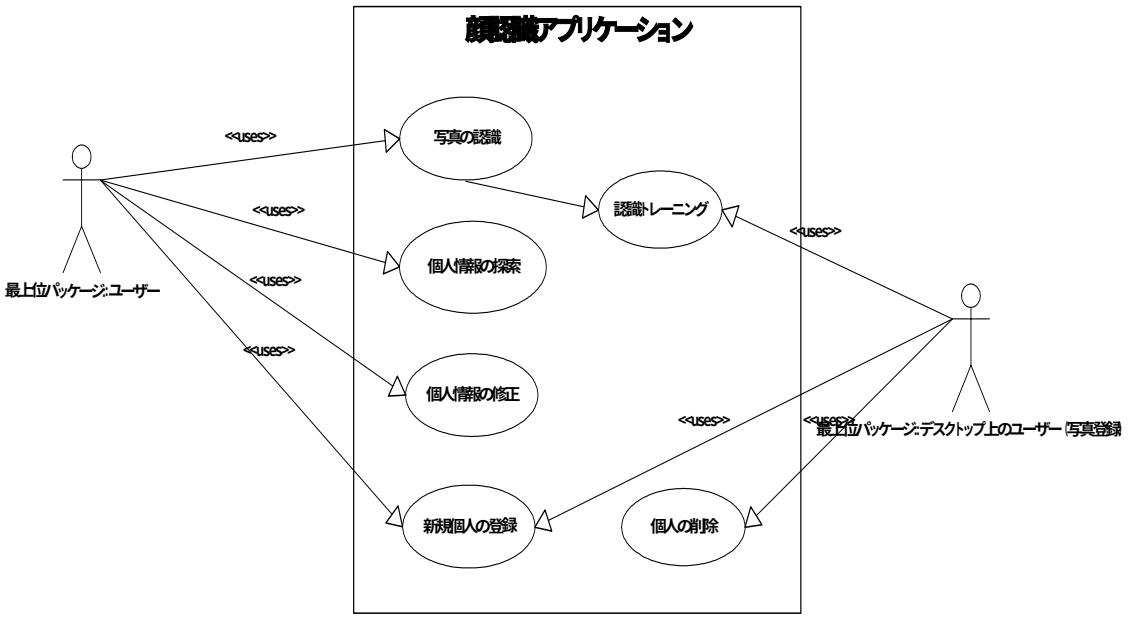
3.3　UML による顔認識アプリケーションの外部設計

3.3.1　要求分析で行う作業

3.3.1.1　ユースケース図の作成

まず、ユースケース図を作成する。ユースケース図というのは外部から見たシステムの業務と関係者を定義するもので、システムが外部に提供する機能単位（ユースケース）と外部の要員（アクター）を記述する。人型のアクターと楕円形のユースケースをシステムの外部と内部に配置することで、一目でシステムの機能と概要を把握できる。

（図 3-2）今回作成するシステムのユースケース図



この作成されたユースケース図のそれぞれの機能単位（ユースケース）について、ユースケース記述を作成する。

3.3.1.2　ユースケース図の作成

ユースケース記述はユースケース図のそれぞれの機能単位（ユースケース）についてどのように動作するのかを詳細に示したもので、この段階でシステムの動作の詳細を定義する。それゆえ、ここで作業をきちんと行うことで後のシステム分析が効果的なものとなる。なお、今回は携帯電話側で利用するユースケースについてのみ限定した。

なお、この段階でユースケースの定義漏れを防ぐため、ユースケース記述では必ず以下の項目を定義するものとする。

概要：このユースケースの機能を簡単な文章で記述する
対応するアクター：このユースケースを利用するアクターを列挙する
前提条件：このユースケースを起動する際に成立するべき前提条件を記述する
処理：アクターとシステムの動作を記述する
後条件：このユースケースを終了したときに成立する条件を記述

（ユースケース記述１：個人の認識）

概要：　　顔認識を行い、該当者について個人情報のいくつかを出力する
対応するアクター：携帯電話側のユーザー
前提条件：　なし
処理：
（１）携帯電話で写真を撮影し、画像を発信する。
（２）サーバは画像を受信する。
（３）サーバは受信した画像を顔認識ライブラリに投入する。
（４）顔認識ライブラリの示した該当者を元に個人情報を検索する
（５）サーバは検索した個人情報を出力する
（６）携帯電話は該当者リストを表示する。
後条件：　認識に利用した画像データが一時保存される
　　データベース上で個人を示すキー3 人分が呼び出し側に保存される
補足＊　　顔認識に時間がかかり顧客を待たせるというのは本末転倒、要配慮

不意に次の来客が現れることに対処するため、どのユースケースからも可能な限り速やかに（１）の顔認識のユースケースに戻れるよう配慮する必要がある。

（ユースケース記述 2：個人情報の検索）

概要：　　指定された個人について詳しい情報を出力する
対応するアクター：携帯電話側のユーザー
前提条件：　「個人の認識」が完了していること
処理：
（1）携帯電話上で該当者リストから候補を選択し、候補を示すキーを送信
（2）サーバは該当する個人を示すキーを受信する
（3）その個人が写真の該当者であるとみなし、認識トレーニングを行う
（4）個人を示すキーを元に個人情報を検索する
（5）サーバは検索した個人情報を出力する
後条件：　データベース上で個人を示すキー1 人分が呼び出し側に保存される

（ユースケース記述 3：個人情報の修正）

概要：　　指定された個人について登録された個人情報を修正する
対応するアクター：携帯電話側のユーザー
前提条件：　「個人情報の検索」が完了していること
処理：
（1）携帯電話上のフォームで個人情報を修正して送信
（2）サーバは修正された個人情報と個人情報を示す主キーを受信する
（3）サーバは DB 内の個人情報を修正し、更新する
（4）サーバは交信に成功した場合、成功した旨を送信する。

ユースケース記述 4：携帯電話上からの新規個人の登録）

概要：　　新たな個人を登録する
対応するアクター：携帯電話側のユーザー
前提条件：　「個人情報の検索」が完了していること
処理：
（1）携帯電話上のフォームで登録するべき個人情報を作成して送信
（2）サーバは登録するべき新たな個人の個人情報を受信する
（3）顔認識用の DB に新規レコードを付加する
（4）個人情報用の DB に新規レコードを付加し、顔認識用の DB 上で作成された主キーを付加する
（5）その個人が写真の該当者であるとみなし、認識トレーニングを行う
（6）携帯電話は成功あるいは失敗した旨を報告する
後条件：　なし

3.3.2 システム分析で行う作業

3.3.1 章で作成したユースケース記述の中から、画面やデータなどアプリケーションの中でユーザーに対応するオブジェクトを抽出します。

3.3.2.1 クラス図の作成

ユースケース記述から該当箇所を引用し、実際に画面と利用するデータを抽出した。それから、その内容を元にクラス図を作成する。

クラスという概念はオブジェクトを記述する単位で、クラス図の作成とはオブジェクトの種類を集約する作業である。具体的にはオブジェクト同士の関係を以下の 3 種類の関係を利用して集約する。

（1） 汎化（△マークの直線） 同類項となるクラスを集約
（2） 依存関係（点線） そのオブジェクトは別のオブジェクトから利用される
（3） 合成（◇マークの直線） オブジェクトを最終的なアプリケーション１つに集約。
（4） 関係（ただの直線） オブジェクト同士の対応（例えば 1 対 1 対応）の記述

（表 3-2）ユースケース記述の内容と抽出されたオブジェクト：画面について

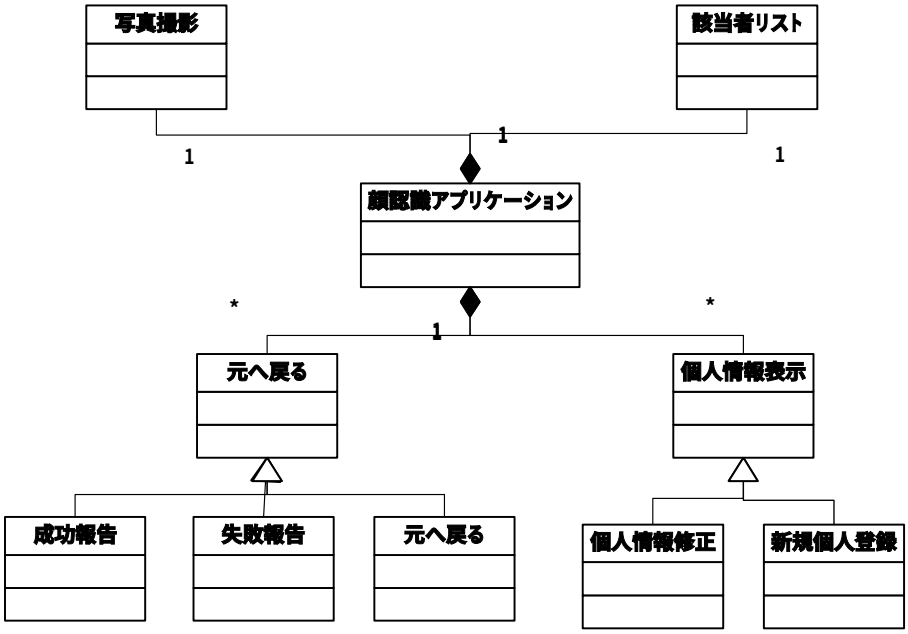
（ユースケース記述１：個人の認識）
処理： （1）携帯電話で写真を撮影し、画像を発信する。 ⇒ 写真撮影画面
（6）携帯電話は該当者リストを表示する。 ⇒ 該当者表示
（ユースケース記述 3：個人情報の修正）
（1）携帯電話上のフォームで個人情報を修正して送信
（5）携帯電話は成功した旨を報告する ⇒ 成功報告画面
（ユースケース記述 4：携帯電話上からの新規個人の登録）
処理： （1）携帯電話上のフォームで登録する個人情報を作成 ⇒ 新規登録画面
（7）携帯電話は成功あるいは失敗した旨を報告する ⇒ 成功画面など

（表 3-3）ユースケース記述の内容と抽出されたオブジェクト：利用するデータ**について**

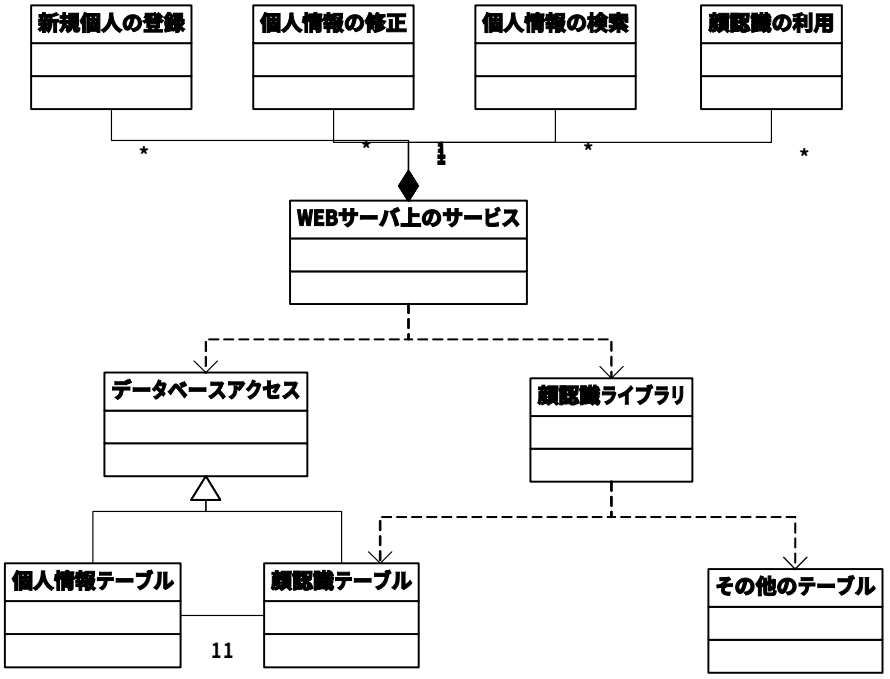
（ユースケース記述１：個人の認識）
処理： （5）サーバは検索した個人情報を出力する ⇒ 個人情報 DB
（ユースケース記述 4：携帯電話上からの新規個人の登録）
処理： （3）顔認識用の DB に新規レコードを付加する ⇒ 顔認識用 DB

また、ユースケース記述の題目にある「個人の認識」、「個人情報の検索」、「個人情報の修正」、「新規個人の登録」の 4 業務は、サーバ上の業務でこれもオブジェクトに相当するので抽出する。同様に、顔認識ライブラリの実行も業務なのでオブジェクトとみなす。

（図 3-3）携帯電話上のクラス図



（図 3-4）WEB サーバ上のクラス図



3.3.2.2　シーケンス図の作成

オブジェクトの担当範囲を明確化し、更にオブジェクト同士の関係を定義するためにシナリオ分析という作業を行う。そのため、実際のユーザーがどのようにシステムを操作するのかモデルケースを作成し、その内容に沿ってオブジェクト同士を関連付ける。

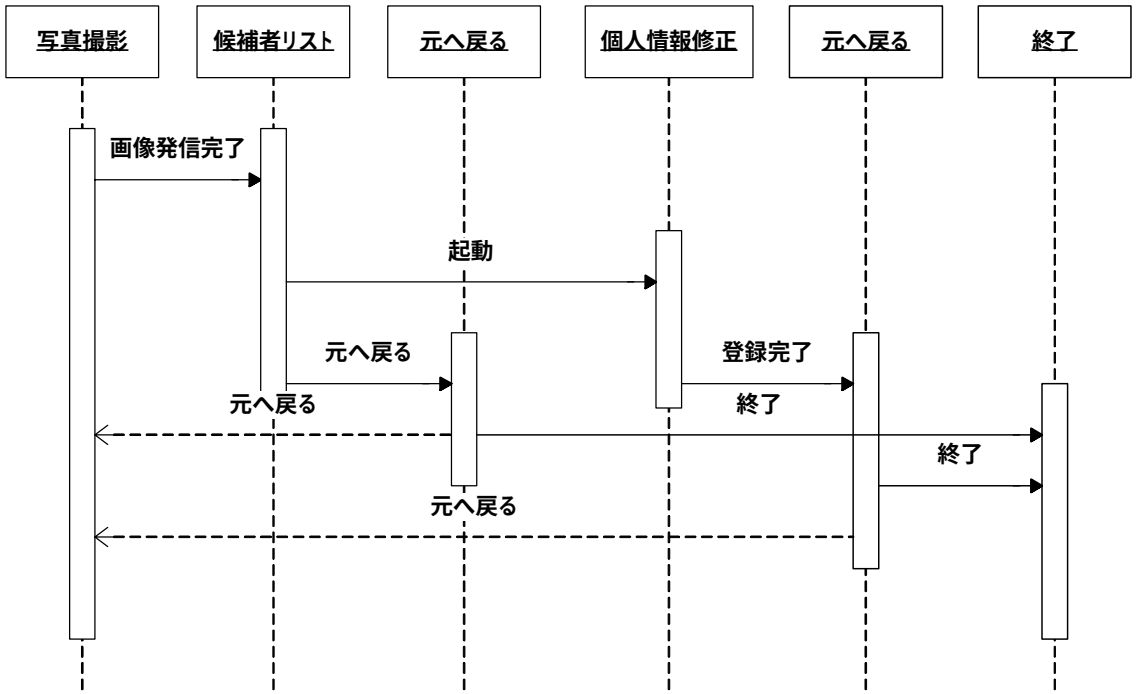
そして、最終的な結果をシーケンス図にまとめました。シーケンス図は複数のオブジェクトの協調動作を記述したもので、四角の部分が前述したオブジェクトの部分に、矢印の部分が操作や送受信されるデータに相当します。今回は画面についてのみまとめました。

（シナリオ 1）

レストランに来客があった。そこで、ある従業員が座席に案内している間に、別の従業員が顧客を撮影した所、その顧客が何者であるのかが候補者リストが示された。そこで、候補者リストの情報から明らかにその来客が何者であるのか分かり、詳しい情報を探索した所、同じチェーンの常連客で好みの情報が予め分析されていた。

そこで、その情報を元に「お勧めメニュー」を紹介したところ、その中のあるメニューが選ばれた。注文の受領後、顧客の注文内容を改めて入力した。

（図 3-5）シナリオ 1 のシーケンス図

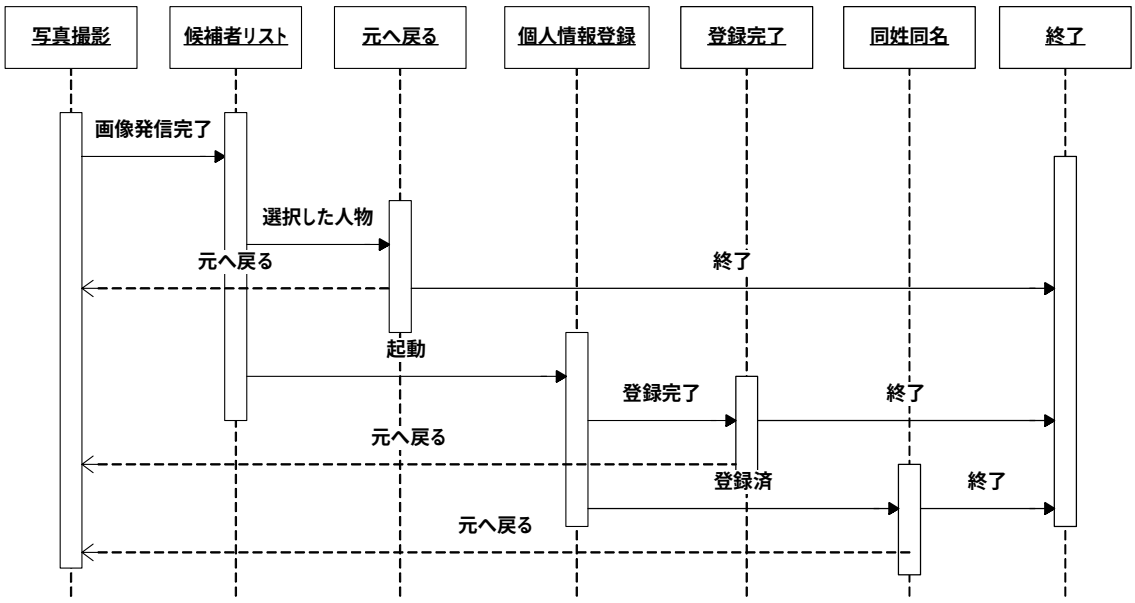


（シナリオ 2）

あるコピー機の修理会社では担当者が割合と頻繁に入れ替わるので、顧客情報をデータベース化して共有している。ある日、修理の依頼があったので担当の従業員が出動しました。そこで、依頼を行った担当者の写真を撮影して彼が何者だったのかを検索しようとした所、該当者が見当たらなかった。

そこで、コピー機の修理が終わり会社へ戻る途中、新たな担当者の情報を入力することにした。登録ボタンを押した所、なぜか同姓同名の全く異なる顔の人間を登録したとの警告が出された。

（図 3-6）シナリオ 2 のシーケンス図



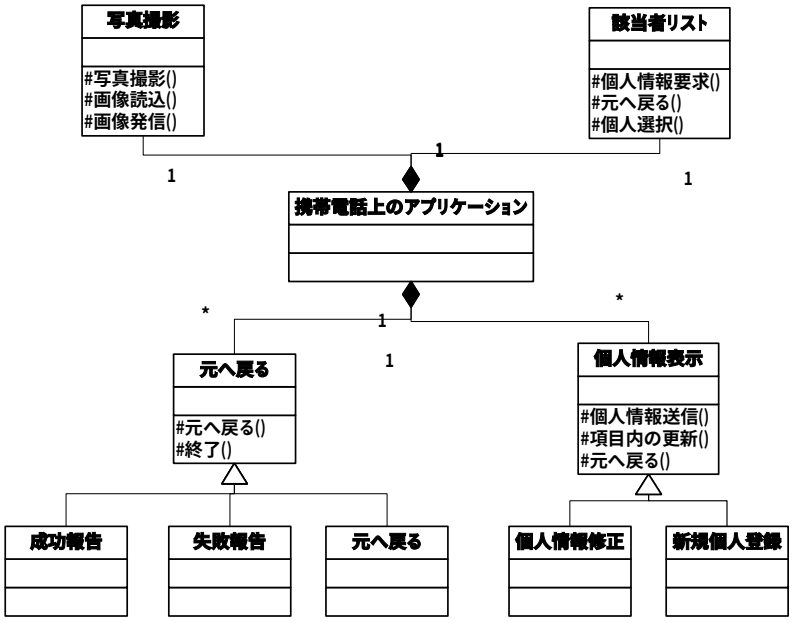
なお、ユースケース記述に記述されているように、このアプリケーションは連続使用を容易にする必要がある。そのため、シーケンス図の作成において、全てのオブジェクトからも 2 操作以内に顔認識（写真撮影）に戻れるよう配慮しました。

また、説明の都合上、本来はオブジェクトではない終了状態も 1 つのオブジェクトとしてシーケンス図上に書き込みました。

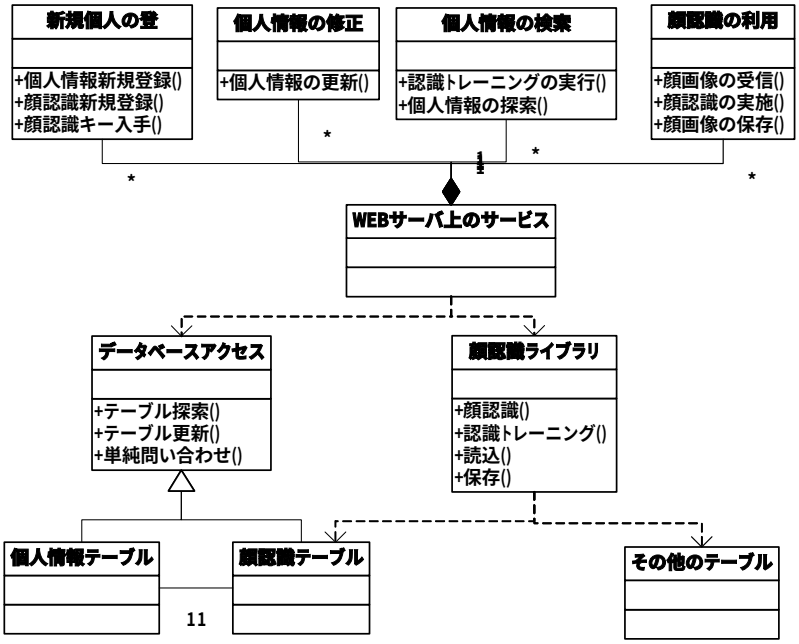
3.3.3　クラス図の詳細化

シーケンス図を作成してクラス図に矛盾のないことを確認した段階で、ユースケース記述に沿ってクラス図のオブジェクトに行うべき動作を記述する。オブジェクト指向の開発では、最終的にこの操作がプログラミングのときに記述する関数（メソッド）となる。

（図 3-7）携帯電話上のクラス図



（図 3-8）WEB サーバ上のクラス図



第 4 章　顔認識アプリケーションの実装

4.1　顔認識アプリケーションに必要な要素技術の習得と利用

以降、しばらくの間、内部設計に必要な、顔認識ライブラリ、SOAP クライアント、WEB サーバといった今回のアプリケーションで用いる主要な構成要素について説明する。

4.1.1　画像処理ライブラリ OPENCV とその具体的な実装方法

4.1.1.1　画像認識ライブラリ OPENCV とは

今回利用する OPENCV（CV というのは Computer　Vision の略）というのは Intel 社がロシアのノブゴロドにある研究所で作成した一連の画像処理ライブラリで、IPL（Intel Primitive　Library）の上位ライブラリで、フリーで提供されている。この IPL というのはインテル社の数値計算ライブラリで、最新版は有料である。COM コンポーネントではない実行形式の DLL ファイルの形で提供されている。

この OPENCV という画像処理ライブラリの中に隠れマルコフモデル（HMM）を利用したパターン認識の機能があり、これを利用すると 2 つの画像の近似度を計算できる。それを応用した顔認識のサンプルアプリケーション HMMDEMO が提供された。

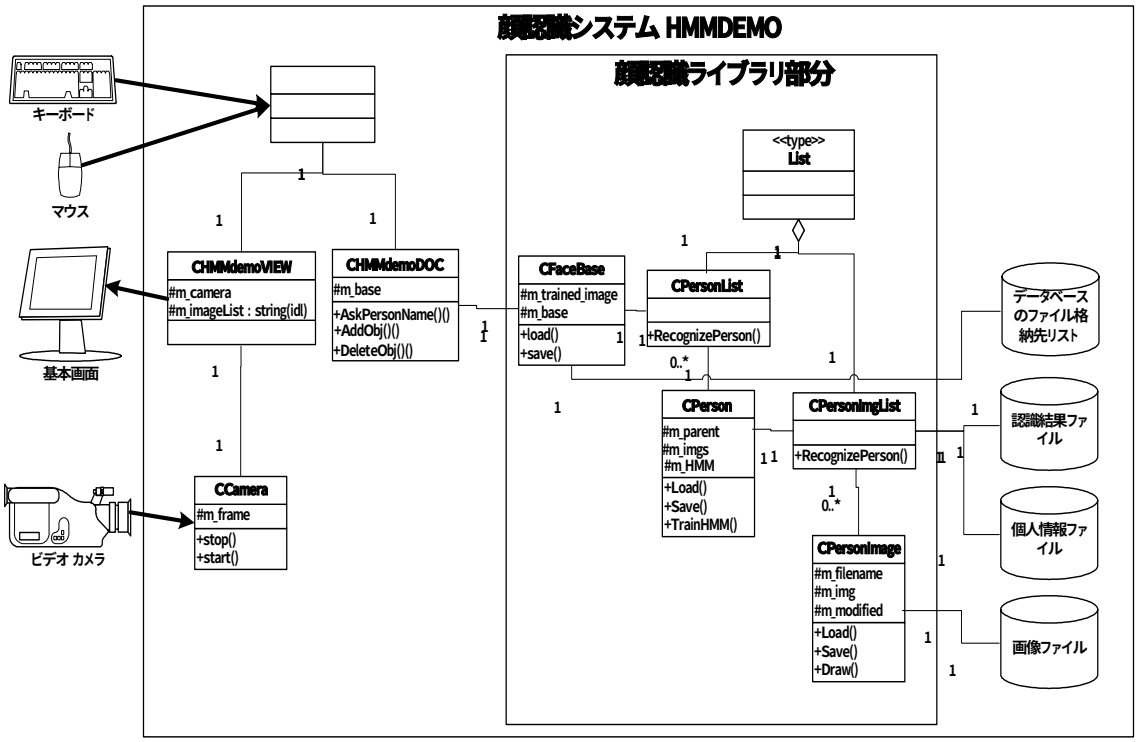
現在、大手電機各社で実用化されている顔認識では、人間工学的な側面も加味してモデルを立てているため性能的に疑問があったが、320×240 サイズの画像で自分と複数の研究室の同僚を識別させた所成功したので性能は十分とみなし利用することにした。

（写真 4-1）顔認識アプリケーション HMMDEMO

4.1.1.2　WEB サービスに対する顔認識機能の実装

この顔認識のサンプルアプリケーションは、図に示したように、WINDOWS 上で機能する単体のアプリケーションであり、MFC の枠組みに沿って構築された図 4-1 のような構造のアプリケーションである。

(図 4-1)顔認識サンプルアプリケーション HMMDEMO の構造



従って、このままでは WEB サービスで顔認識機能を利用することは不可能である。そこで、図 4-1 で示したアプリケーションのうち、顔認識機能を担った右の四角の部分を切り出し、WEB サービスで利用可能なプログラム部品に加工する

具体的には以下の作業を実施した。

（1） アプリケーションのうち顔認識機能を担う四角で囲まれた部分に相当するソースコードを切り出す
（2） ソースコードのうち、ファイル入出力の部分をデータベースの利用に改める
（3） COM コンポーネントの形にまとめ、実際に機能するかテスト
（4） NET、COM 混在オブジェクトに書き直し、WEB サービスから利用可能な形に作成した COM コンポーネントを書き直す。

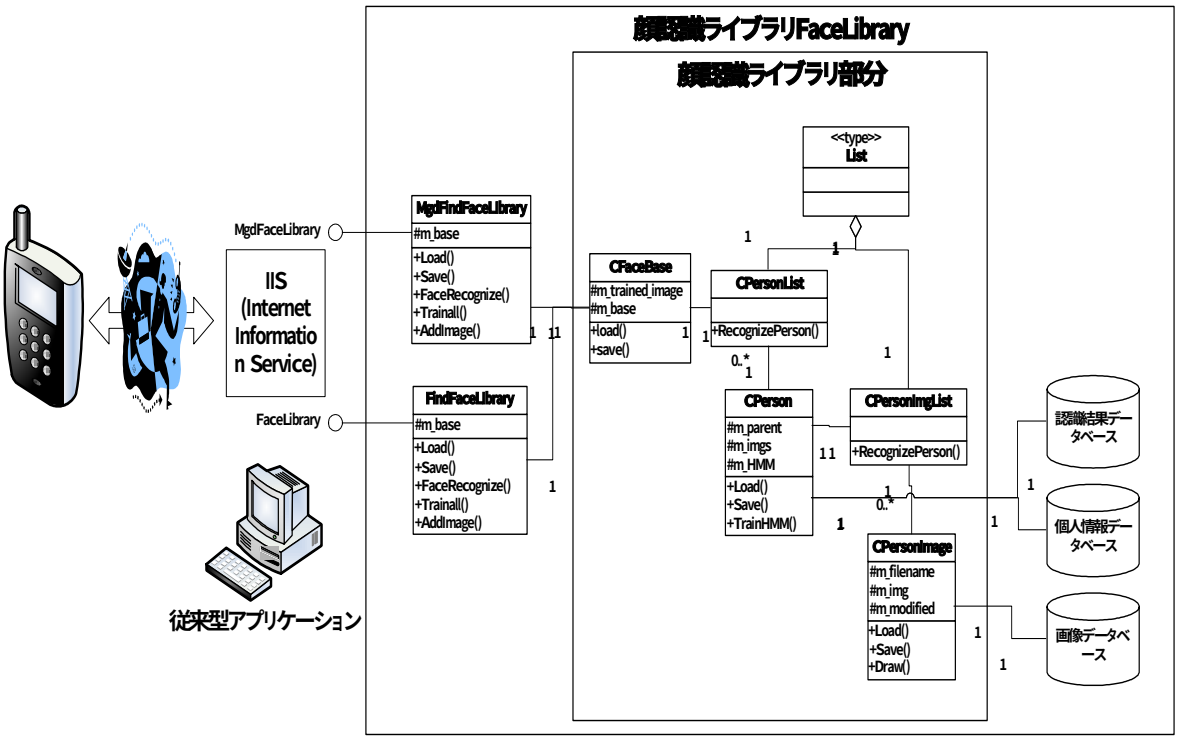
WEB アプリケーションは複数ユーザーの利用を前提としているため一人のユーザーがあるファイルを利用している時には別のユーザーにファイルを使わせない同時実行制御という機能が必要である。同時実行制御のプログラミングは困難でかつデータベースソフトが同時実行制御について充実した機能を提供しているので、アプリケーションの方を通常のファイル利用からデータベースアクセスに改めた。（文献[8-2]参照）

また、WEB サービスは ASPNET から提供されるので、作成された COM コンポーネントを NET、COM 混在オブジェクトに書き直す必要があった。ASPNET の WEB ページ（人間用）では COM コンポーネントを直接利用できるが、WEB サービス（機械用）では COM コンポーネントは直接利用できないからである。

結果、最終的に図 4-2 で示した構造のライブラリ FaceLibrary を作成した。このライブラリを VBNET で作成したサンプルアプリケーションで動かした所、完全にサンプルアプリケーションと同様に機能させることに成功した（写真 4-2 参照）。

また、測定を行った結果、認識速度は十分であることが確認できた。表 4-1 に BMP と JPEG とあるのはデータベース内部の画像保存方式で、圧縮すれば認識速度が遅くなる関係が如実に示されている。なお、最終的に JPEG（圧縮）方式によるデータ保存を選択した。

（図 4-2）顔認識ライブラリ FaceLibrary の構造



（写真 4-2）単体のアプリケーションでの FaceLibrary の利用

（表 4-1）顔認識ライブラリのスペック（数字は全て秒単位）

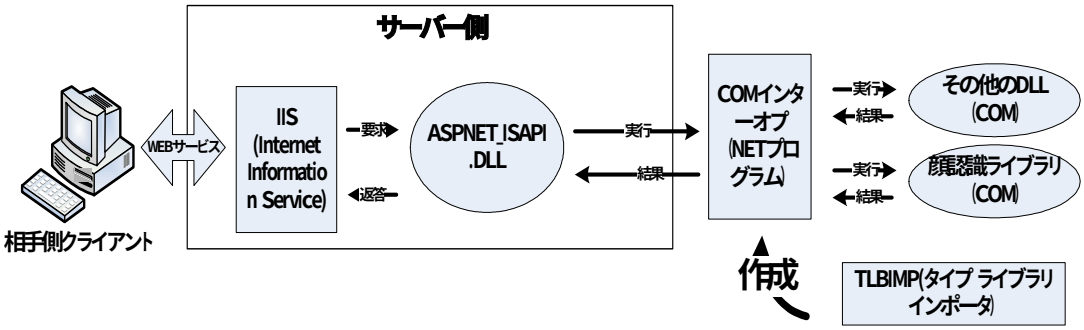
	JPEG	BMP	備考
LOAD（画像なし）	0.35	0.35	
LOAD（画像あり）	1.64	1.03	（画像10枚あたり）
顔認識	0.3	0.3	（候補6人あたり）
認識トレーニング	2.4	11.14	（画像30枚あたり）
結果保存	0.4	0.4	（画像なし）
顔認識まで合計	0.83	0.83	
最終作業合計	2.64	2.04	

ただ、VBNET で NET フレームワークによるサンプルアプリケーションから機能したにも関わらず、ASPNET の提供する WEB サービスから動作させたところ機能しなかった。「依存関係にあるファイル（画像処理ライブラリを構成するファイル）が見つからない」と、NET の仮想マシン（CLR）は実行を拒否した。

ASP.NET の WEB サービスは既存のシステムを WEB サービスに統合するから存在価値がある。既存のシステムはほとんどが 4.1.3 節で後述する ADO（ActiveX Database Object)、または ASP を利用した COM ベースのプログラムで構成されている。従って、ASP.NET には COM を利用する機能が存在しているはずである。

調査したところ、TLBIMP（テーブルライブラリインポータ）と呼ばれるツールを利用して相互運用アセンブリという翻訳プログラムを作成すればCOMで記述されたプログラムをNET 環境でも利用可能であることがわかった。この方法は Microsoft 社も推奨していた。それゆえ、MFC 形式のもとのプログラムを比較的書き換えの容易な COM 形式のライブラリに書き直した。以上に述べた理由でこの判断は現在も正当であると考えられる。

（図 4-1）TLBIMP と COM のイメージ



今回は、OPENCV というライブラリ、COM 形式ではない旧式の DLL ファイルの集合も利用する。そこで、COM の利用ではなく旧式の DLL ファイルの利用が問題の原因ではないかと考え調査したところ、2 つのことが分かった。

（1） PINVOKE という方式が存在し、最悪でもある種の翻訳プログラムを作成すればNET から直接旧式のライブラリを利用できること、

（2） VC++の場合、直接添付されたインクルードファイルをインクルードすれば、ほとんどの業務を NET 側でこなしそのまま旧式のライブラリを利用できること

最悪の場合、プログラムを NET 形式に書き換えればよいことが分かった。しかし、いずれにせよ、OPENCV には COM で作成した部分からアクセスするだけで、NET 部分はあくまでインターフェースとして利用しただけである。そうであるからにはしかるべき構成ファイルの登録方法があるはずである。

実際、ASP.NET の WEB サービス本体も含め、NET 環境のプログラミングでは XML で記述した環境ファイル（ポリシーファイル）が比較的大きな地位を占め、定数やアクセス権限、必要なファイルの記述を担当している。しかし、調査した範囲では原因となるような箇所は見つからなかった。

現在､複数のソースから COM+という COM のインターネット応用技術についてもう少し調査するべきであるということが分かり調査中である。

4.1.2 携帯電話上で実際にシステムを構築するにあたっての課題

4.1.2.1 軽量な SOAP クライアントの入手

今回、携帯電話 WEB サービス技術を利用するが、そのためには SOAP クライアント機能を提供するライブラリを入手し、自分のプログラムに付加する必要がある。

しかし、SOAP クライアントに必要不可欠な XML パーサーは、例えば WINDOWS 標準の MSXML の場合、実に 10MB に及ぶ。携帯電話でそうした XML パーサーを利用できるわけがないので、私は別の場所から JAVA で動作する極めて軽量な SOAP クライアントを入手した。それが KSOAP である。

KSOAP というのは米国の有志が組み立てた JAVA 上の軽量な SOAP クライアントで、元々は米国で主流の Palm 用の SOAP クライアントとして開発された。この技術を日本で携帯電話に応用した所成功し、携帯電話用の SOAP クライアントとして認知された。

4.1.2.2 携帯電話で利用可能な SOAP クライアントの作成

入手した KSOAP のソースコードを元に、NTTDOCOMO の携帯電話用の SOAP クライアントを作成した。KSOAP は J2ME によって一応携帯機器用に設計されているが、携帯電話での利用についてまでは配慮されていないためいくつかの改造を必要とした。

具体的には主に以下の作業を必要とした。

（1）通信部分の書き直し

KSOAP の配布版の通信プログラムは当然ながら NTT の携帯電話の規格に合致しておらず、携帯電話で利用可能にするため全面的な改造を必要とした。

（2）BASE64 処理（バイナリ符号処理）部分の追加

KSOAP でバイナリデータ（画像データ）を送信するには BASE64 と呼ばれる符号化の手段が必要だが、携帯電話用の JAVA には装備されてないのでプログラムを入手し、利用可能なようにする必要があった。

（3）不必要な部分の削除

携帯電話では自前でライブラリを設置できないので、プログラムは全て一箇所にまとめて持ち込む必要がある。そのため、依存関係を全てのプログラムにわたって分析して検出し、当座の業務に不必要なクラスライブラリは全て削除した。

結果、合計サイズが 48KB の SOAP クライアント作成に成功した。クラスライブラリレベルでなく関数レベルで不要部分を削除すればサイズをもう少し減量できると考える。だが、いずれにせよ、現在の携帯電話で利用するにはまだまだ重過ぎるので、テストの際には利用しても、実用化の際には HTTP 通信部分を除いては利用を取りやめ、必要な WEB サービスのやりとりは自分で直接プログラミングすることになる。しかし、十分軽量であり、今後携帯電話の容量が多少でも増大すればプログラムに組み込み可能と考えている。

4.1.2.3 通信の暗号化に関する考慮

人間の顔写真と関連する氏名などの個人情報は、写真付きの個人情報なので、取扱には重大な注意が必要である。そのため、今回利用する SOAP クライアントでは暗号化によるセキュリティの方法を考慮する必要がある。

携帯電話でセキュリティを確保するには現在 2 通りの手段がある
（1）HTTPS の利用　　NTT の携帯では Verisign 社などの電子証明書を利用する
（2）暗号ライブラリの導入　　JAVA の暗号ライブラリを導入する

（1）の手法は赤外線通信と組み合わせ携帯電話でクレジットカードの決済を行うサービスでも利用されている（文献[7-5]を参考）。比較的容易に実装可能でしかも信頼性の比較的高い手段である。しかし、いくつかの問題をかかえる

（1） HTTPS を利用した場合、画像を含め全通信が暗号化される。画像データは暗号化しても無意味な上、巨大なので通信時間を大きく消費する
（2） HTTPS はあくまで通信路の暗号化であり、SOAP 規格で SOAP メッセージを中継するサーバで問題点があるとの指摘がある。（文献[2-2]を参考）

NTT の携帯電話では今の所 HTTP と HTTPS の使い分けは出来ない。後述するように時間内にアプリケーションを実行できないのは致命的なので、HTTPS を利用した場合（1）の通信時間の問題は重大な欠陥になりかねない。それゆえ、（2）の暗号化による方法を採用するべきであると考えた。

そこで、JAVA の標準暗号化ライブラリ JCE（Java Cryptography Extensions）を利用して暗号化機能を組み込む方法を考えた。しかし、暗号関係のプログラム自体で KSOAP に匹敵する記憶容量が必要であることが分かった。チュートリアル用のプログラムを入手して改造することも考えたが暗号計算に大幅に実行時間を取られるのではないかという疑念があった（携帯機器用の軽量高速な暗号化ライブラリは現在注目されている技術のひとつでそれ自体が高価な商品である）。そういう事情が背景にあったことと開発の時間的な問題もあり、今回は一切暗号化は実装していない。

4.1.2.4　セッション管理に関する考慮

4.1.2.4.1　顔認識における認識トレーニングの必要性

顔認識アプリケーションは 2.4.1 節で前述したように予めパターン情報と顔画像から得られた情報を照合して顔認識を行う。そのため、単純に顔画像を送信してその結果を送り返すだけでなく、顔画像を登録し、人名と共に顔認識アプリケーションに入力してパターン情報を作成する（認識トレーニングする）必要がある。

しかし、パターン情報の作成にはこれまで登録した顔画像全てをデータベースから読み出すので、表 4-1 にあるように相当な時間を消費する。これは「顔認識に時間をかけない」というユースケースに記述された要求（3.3.1.2.節）に反する。また、顔画像が誰のもののであるのか報告を受けなくてはそもそも「学習」しようもない。ゆえに、顔認識と認識トレーニングは別々のプログラムで行う必要がある。

そのため、3.3 節で作成されたシーケンス図（図 3-5、図 3-5）内では「個人情報の修正」、「新規個人の登録」という作業を行うため選択された候補を該当者と見なし、プロファイルの送信と連動して認識トレーニングする方式を採用した。

問題は、顔認識の実行と認識トレーニングは別々のプログラムで行われることである。

4.1.2.4.2　セッション制御に関する調査

通常、WEB サービス技術では 1 回関数が呼ばれたら、その実行結果はデータベースに出力しない限り保存されることはない。WEB アプリケーションは前述したようにホームページ規格の延長であるためである。

しかし、今回のようにすぐ別の関数を呼び出す場合、それでは問題である。そのため、ホームページにも Cookie 技術が利用できたようにセッション制御と呼ばれる方法で特定のユーザーに対する実行結果を保存する。

ただし、Cookie 技術はクライアント側にデータを保存して接続するたびにデータを持ち寄る技術であり、携帯電話では利用できないとマニュアルに明記してある。そこで、ASP.NET でサーバ側にデータを保存する技術について調査した。結果、以下の２種類の技術が存在することが分かった。

（１）Cookieless 技術　　メモリに保存、相手の URL をユーザー識別に利用
（２）データベースへの保存　　データベースに保存

可能な限り高速であることが望ましいので、データをメモリかメモリに準じた場所に保存する Cookieless 技術の利用が望ましい。しかし、相手の URL をユーザーごとのデータ識別に用いると ASP.NET のマニュアルに明示された。

携帯電話の場合、この URL は携帯電話会社の携帯電話回線とインターネットを結ぶゲートウェイサーバの URL となり、直接個々の携帯電話の識別に利用できない（注:2003 年 10 月現在の話、会社ごとによっても事情は異なる）。そこで、課金に用いられる携帯電話の固有識別番号を識別に利用するのが最適だが、携帯電話の固有番号は HTTP の環境変数で送信されるのは確かだが具体的に利用する方法が見つからなかった。それゆえ断念した。

結局、携帯電話の固有識別番号を 2.3.3 節で前述した SOAP ヘッダで受信し、データは固有識別番号をキーにデータベースに格納する方法で保存する方法を利用した。SOAP ヘッダはデフォルトでは利用できないので KSOAP と ASP.NET の WEB サーバ双方を改良した。

テキストデータでの実験を実施し、画像データでも実験成功まで後一歩まで迫った。しかし、時間不足のため、今回の顔認識アプリケーションにこの機能は実装していない。

また、SOAP ヘッダには携帯電話識別番号と事前設定済みのパスワードも設定したので、これを利用してユーザー認証を行うことが可能である。顔認識アプリケーションは個人情報を取り扱うのでインターネット上で運用する場合必要不可欠である。ただし、実際に利用する場合は 4.1.2.3 節で前述した暗号化の手法を利用し、パスワードを暗号化するべきである。

前述した理由で携帯電話の URL は事実上認証に利用することが不可能である。また、いちいちログイン名を手入力するのではユビキタスの基本理念に反する。それゆえ、4.1.2.3 節で前述した SSL を利用する方法（携帯電話に内蔵された SSL の証明書の利用）と並び、携帯電話の固有番号と SOAP ヘッダを利用した認証手法は携帯電話でユーザー認証を行うひとつの手法に数えられる。

（写真 4-3）セッション技術の応用（テキストデータにて）

4.1.2.5　送信する写真サイズの決定

今回の論文では顔認識を携帯電話で実現しようとしている。そのため、顔の画像を携帯電話から送信する必要に迫られ、送信する顔写真のサイズを決定する必要がある。

顔写真のサイズが大きければ大きいほど認識の精度は向上する。携帯電話の場合、悪条件での撮影を強いられるのでこのことは重要である。しかし、顔写真のサイズが大きければ大きいほど通信時間や料金がかかる。3.3.1 節で記述したように顔認識のため数秒も人を待たせるというのは大変問題である。そのバランスをとらなければいけない。

写真のサイズとファイル容量は表 4-2 で示したように正比例する。この表 4-2 を見る限り、携帯画面サイズの写真の送信であれば、AU の携帯電話であってもほとんど時間がかからないことが分かった。繰り返しになるが顔認識アプリケーションで実行時間を費やさない利点は決定的である。

(表 4-2)画像のサイズと送信時間の関係（文献[7-1]より作成）

	圧縮前	圧縮後	送信時間（144kbps 時）
320×240（QVGA）	約 225KB	約 15KB	約 0.96 秒
640×480(VGA)	約 900KB	約 60KB	約 4 秒
1280×960(SXGA)	約 3.6MB	約 240KB	約 16 秒

（注）圧縮は JPEG 規格を利用、圧縮率はデジタルカメラのそれを参考

では、320×240（QVGA）サイズの画像を利用して認識アルゴリズムは大丈夫なのかという疑問がある。これについて、今回利用する OPENCV（INTEL 社の公開画像処理ライブラリ）とそのサンプルアプリケーションを利用して評価した結果、このサイズの画像ならまず大丈夫であろうと言う結論が得られた。

そうなると気になるのが通信料金の問題であるが、パケット通信料金は現在 1 パケット（128 バイト）あたり 0.2 円であり、320×240（QVGA）サイズの画像で構わないので 1 回の認識に約 30 円前後を費やすことになる。この金額の評価は人によって異なるが、ビジネスであれば十分採算が取れる問題のない金額ではないかと考える。

ただし、実際にはこのサイズで送信を試みると携帯電話の通信規格（1 回の通信で送信データは最大 10KB 以内）に抵触するため不可能であった。複数回の通信に分割すれば良いのだが、通信の分割を WEB サービス規格で行うためには前節 4.1.2.4.2 で説明したセッション制御という技術を必要とする。今回はセッション制御技術を実装できなかったので画像サイズの方を縮小するしかなかった。

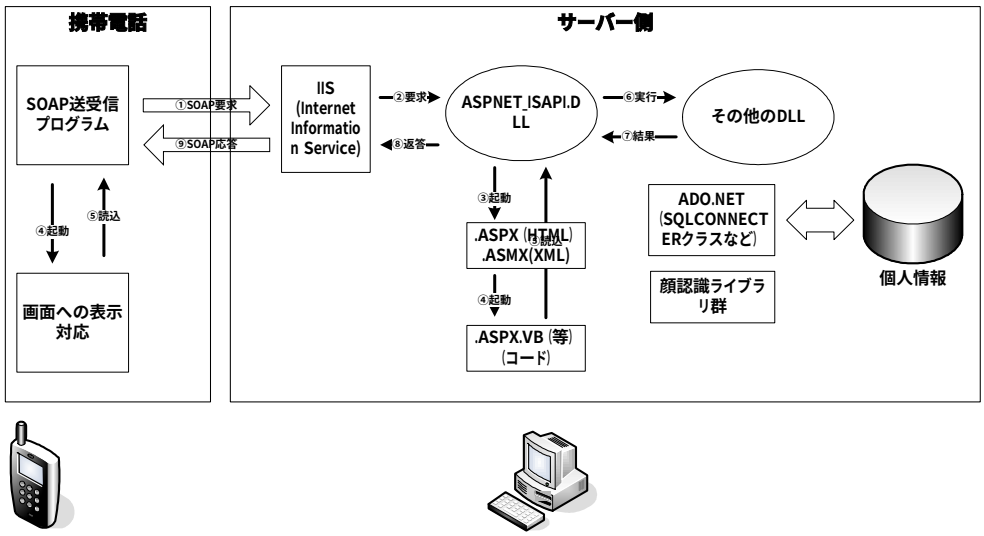
結局、送信する写真のサイズは 192×144 にすることに決定した。

4.1.3 ASPNET と ADONET について

今回利用する WEB サービス用の WINDOWS 側のサーバに ASPNET を選択した。ASPNET は NET フレームワークで提供される Web アプリケーション用のサーバとクラスライブラリ群の総称である。Microsoft 社が提供する WEB サーバ IIS に接続して機能する補助サーバである。（文献[6-7]より引用）

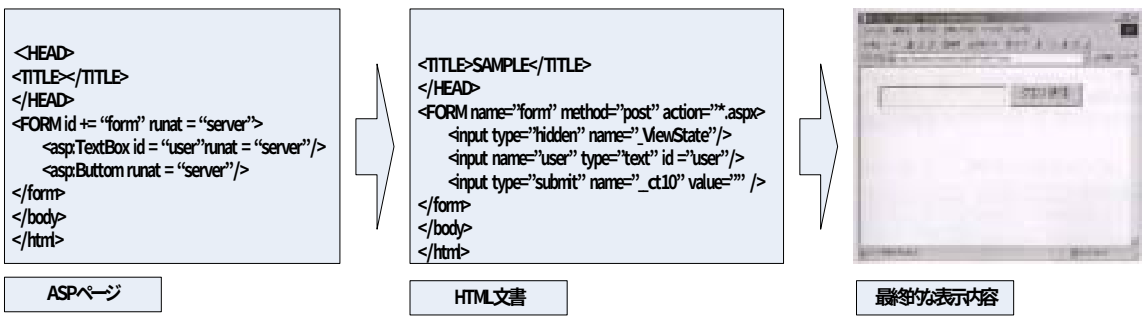
ASPNET は Windows サーバ上でダイナミックな Web ページを動作させるための標準的な基盤とみなされ、Web アプリケーションや Web サービスの開発・構築に用いられる。顔認識ライブラリが WINDOWS 上のプログラムなので ASPNET を選択した。

（図 4-4）ASPNET による WEB サービスの概念図



位置付けとしては従来の ASP の後継にあたる。基本的には ASP と同様、HTML 文書内部に ASP 形式の命令やタグを埋め込み、IIS を通じてサーバにアクセスされた時、その内容を HTML 文書に変換することで機能している。

（図 4-5）ASPNET のプログラム上のイメージ



ASPNET は ASP と比較した場合、以下のような利点がある

（1）HTML 文書とプログラムを分離可能にした

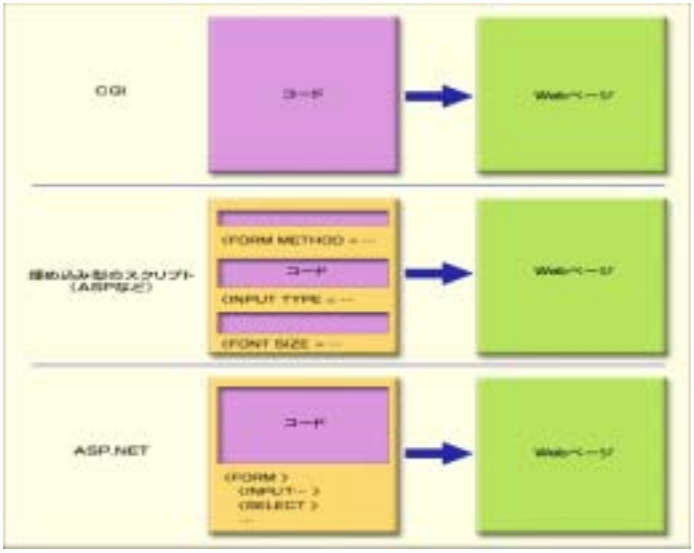
（2） 携帯電話など様々なプラウザの差異を吸収し、相手のプラウザに意図したとおりの文書を表示する

（3） WINDOWS 上で WEB サービスを提供する

ASP は基本的にスクリプト言語で利用し、HTML 文書内部にプログラムを書き込む必要があった。これはプログラムの生産性を著しく妨げた。これに対し、ASPNET は HTML 文書とプログラムを分離し、プログラム部分は本格的な言語で記述できるようになった。

また、ASPNET は基本的に ASP 形式の命令を解釈して HTML 文書に変換するサーバなので、変換プログラムを予め用意すれば様々な形式の HTML 文書を出力できる。その性質を利用して、携帯電話などプラウザの差異を吸収し、相手のプラウザに意図したとおりの文書を表示することが可能になった。WEB サービスの提供もその延長線上にある。

（図 4-6）従来のソフトウェアと ASPNET の関係：プログラムと HTML が完全分離可能！



（文献[4-6]より引用）

今回は ASP.NET を利用するのでそのデータベースアクセス用ソフトウェアとして ADO.NET を採用した。ADONET は Microsoft 社が提供する、Microsoft .NET 環境でデータベースアクセスを行うための基盤となるソフトウェアで、ASPNET で行われる WEB サービスに必要となるためである。

ADO.NET は ADO（ActiveX Database Object）の NET 版であり、NET フレームワーク化したこととネットワーク対応の改良が施されたため、プログラミングが容易になりかつ機能も向上した。代表的な改良点としてデータセットという概念の採用とこれによる非接続型データベースアクセスが挙げられる。

4.2　UML を利用したアプリケーションの内部設計

以上の説明で、アプリケーションの構築にどのような構成要素を利用するべきか概略が示された。そこで、外部設計の段階で作成されたクラス図とシーケンス図、それと資料を元にアプリケーションの本体を設計する。

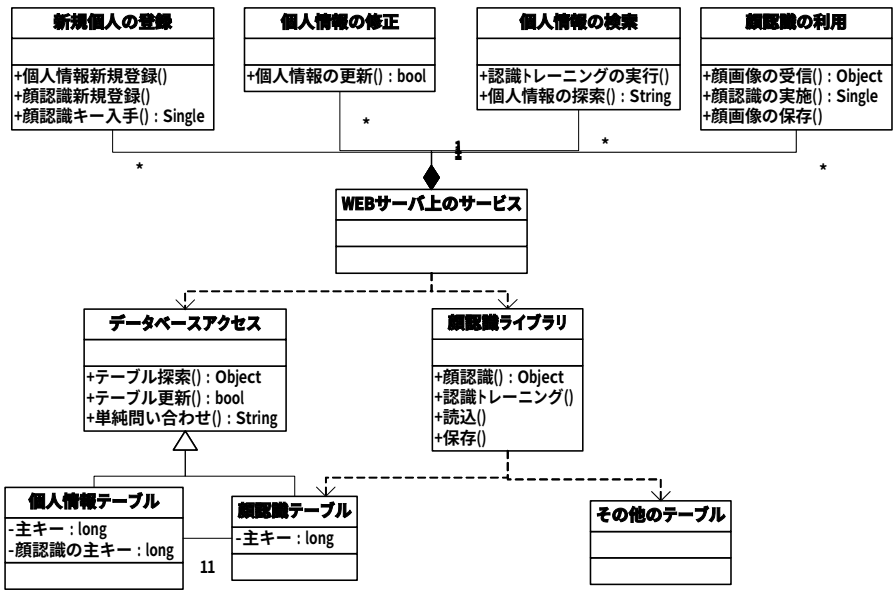
4.2.1　クラス図の実体化

クラス図内のオブジェクトの操作（メソッド）について、返還する値を、シーケンス図やユースケース記述を参考に、実際の環境で利用されるデータ型で定義する。また、この段階でメソッドの公開非公開（+：公開、#-：非公開）も決定し、名前も実際の環境で利用する関数名に切り替える。今回は理解の都合上、最後まで名前は変換しなかった。

4.2.1.1　WEB サーバ上のアプリケーションの場合

WEB サーバ上のアプリケーションは NET フレームワークでプログラミングされる。よって、NET で利用されるデータ型（String、Byte、BOOL、Object など）を利用する。
なお画像は Byte 型の配列で受信されるが、利用する際には画像専用のデータ型に変換する。そのため、図 4-6 では Object 型（何かのクラス型）として取り扱う。

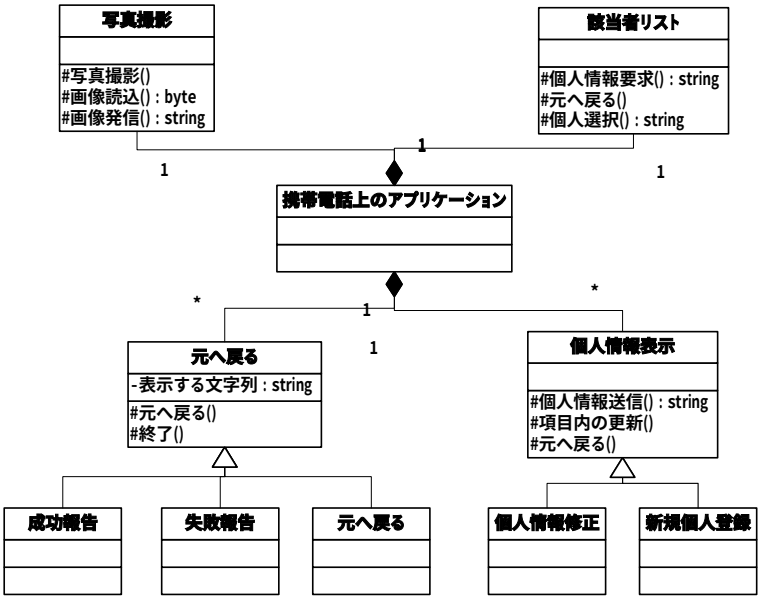
（図 4-7）WEB サーバ上のクラス図



4.2.1.2　携帯電話上のアプリケーション

NTTDOCOMO の携帯電話は JAVA 環境で動作するので、JAVA のデータ型を利用する。今回は技術力の限界で XML パーサーに全ての解析を行わせることが出来なかったので、データの受信関係は全て文字列でデータを受信し、自前で解釈するものとする。

（図 4-8）携帯電話上のクラス図



今回、この段階のメソッドで明白に必要と思われるオブジェクトの所有するデータ（属性）も記述した。その公開非公開、データ型はメソッドのそれに準ずる。

4.2.2　コラボレーション図とパッケージ図の作成

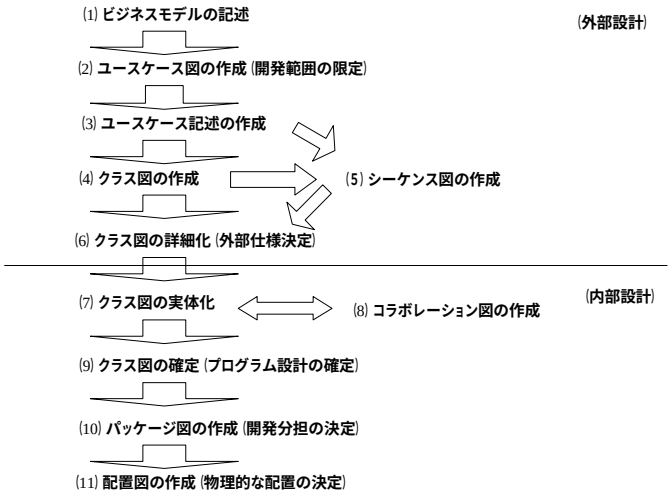
本研究のような個人によるプログラミングでは利用しないが、大規模なソフトウェア開発では以降も行う必要な作業がある。

まず、クラス図を実体化した後、コラボレーション図と呼ばれる実際のプログラム上でオブジェクトの生成消滅までの詳細な動作を示す図面を作成し、この図面と矛盾しないようプログラム内部で利用するメソッド全てを定義する。また、同様に、ユースケース記述と作成されたメソッドを基にして、オブジェクトの属性も全て定義する。更に、依存関係（利用関係）を元にクラス図をいくつかの単位に分割し、パッケージ図を作成する。これで、複数の開発者がいる場合の開発区分を定める。そして、最終的に区分されたパッケージがどこのハードウェアに所属するかを定めるため、配置図を作成する。

実際のプログラミング（実装）は、最終的に作成されたクラス図の属性とメソッドを枠組みとして、その枠内を埋めてゆく形で行う。こうした図面を読み込み、直接プログラムの一部を作成する CASE ツールを利用することも多い。

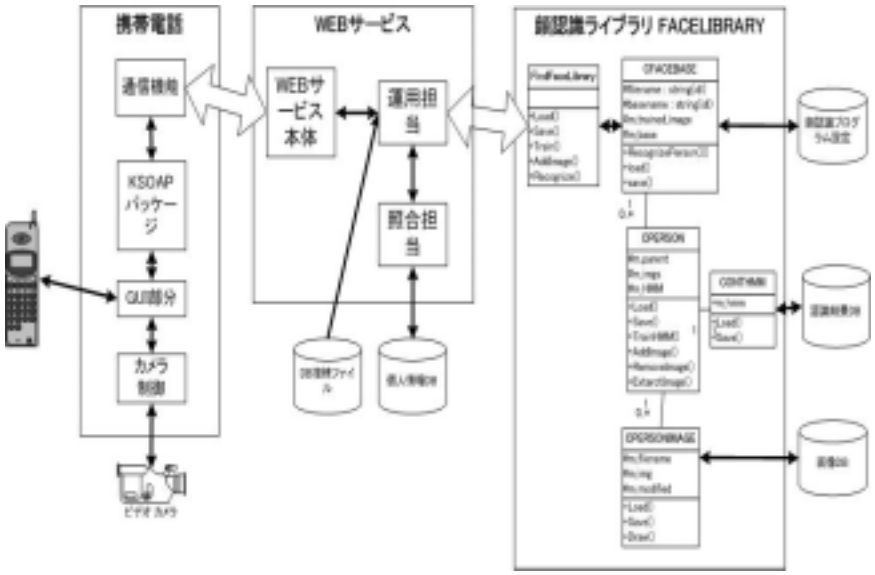
今回は技術を習得しながらプログラムを下から組み立てたので、コラボレーション図を利用してクラス図を完全なものにする作業はしなかった。同様に今回は個人が開発しているので開発区分を定める必要がないので、パッケージ図は作成しなかった。また、携帯電話というハードウェアの限界上、何の機能をどう分担するかは自明に定まってしまうので配置図も作成しなかった。

（図 4-9）UML で行うべき作業（まとめ）



最終的に、既に設計開発が完了した顔認識ライブラリと合わせ、図 4-9 のような構造のアプリケーションを構築することとなった。

（図 4-10）顔認識アプリケーションの構造（概略）



4.3　顔認識アプリケーションの動作確認

これまでの調査や設計を元に、実際にアプリケーションを作成した。結局、図 4-10 に示したように、完全に WEB アプリケーションは構築できなかった。そのため、ビジネスモデル図（図）にあるようにアプリケーションを稼動させることは出来なかった。しかし、顔認識アプリケーションに必要なプログラムの大半の実装に成功した。

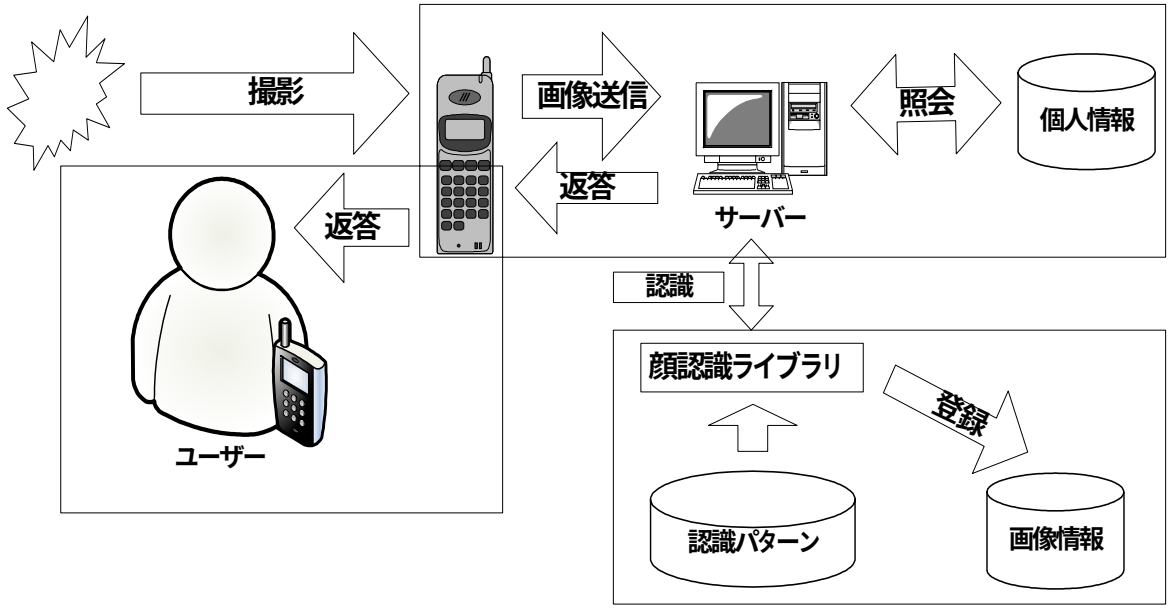
<u>完成した機能</u>

（1） 携帯電話とサーバ間の WEB サービス運用
（2） 顔認識ライブラリ本体
（3） 携帯電話上の GUI

<u>完成しなかった機能</u>

（4） 顔認識ライブラリと WEB サーバの接続
（5）撮影機能（作成したがエミュレータでは検証不可能）

（図 4-11）完成した部分（図 1-4 と内容は全く同様）



そこで、顔認識アプリケーションの完成した一部、特に GUI が設計した通りに機能するのか評価を行った。そのため、ユースケース上の「個人を識別する」「個人情報を修正する」「新規個人を登録する」の 3 機能について、実際に動作させ、ユースケース記述と照合した。ただし、この実験は実機ではなくエミュレータ上で行った。また、この行った実験はあくまでローカル環境内の実験であり、インターネット環境での実験ではない。その点は混乱の原因となるのであらかじめ明言する。

4.3.1 「個人を識別する」機能の動作

画像データが送信され、その結果、個人（私）を識別する。（ただし、エミュレータなので実際の撮影は出来ない。また、今回は、調整がうまく行かず、顔認識機能を WEB サービスに接続することは出来なかった。）

（写真 4-4）携帯電話側の動作

（写真 4-5）データベースの内容を見ると

dbo_INFORMATION：テーブル

INFORMATION	PERSONID	氏名	所属	関係	住所	電話番号
1	1	Iiduka	College	himself	apart	
2	2	Kawanishi	Graduate	Senior	apart	
3	3	Kitamura	College	colleague	apart	
4	4	Prof.Sato	College	teacher	house	
5	5	nobody	none	none	none	
6	6	Tsai	Graduate	Senior	apart	

（写真 4-6）送信された SOAP ヘッダを見ると

```
- <SOAP-ENV:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/">
  - <SOAP-ENV:Body SOAP-
      ENV:encodingStyle="http://schemas.xmlsoap.org/soap/
    - <FaceRecognize xmlns="http://tempuri.org/"
        id="o0" SOAP-ENC:root="1">
        <image xsi:type="SOAP-
          ENC:base64">/9j/4AAQSkZJRgABAQEAYABgAAD/2
      </FaceRecognize>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

4.3.2 「修正した個人情報を送信する」機能の動作

修正画面に入り、実際に登録内容を修正した。そして、その結果を送信した所、無事、登録内容の変更が行われたとの報告があった。では、その通りデータベースが変更されているか確かめた所、こちらも入力したとおりに変更されたことが確認された。

（写真 4-7）携帯電話側の動作

（写真 4-8）データベースの内容を見ると（実行前）

dbo_INFORMATION：テーブル

INFORMATION	PERSONID	氏名	所属	関係	住所	電話番号
1	1	Iiduka	College	himself	apart	
2	2	Kawanishi	Graduate	Senior	apart	
3	3	Kitamura	College	colleague	apart	
4	4	Prof.Sato	College	teacher	house	
5	5	nobody	none	none	none	
6	6	Tsai	Graduate	Senior	apart	

（写真 4-9）データベースの内容を見ると（実行後）

dbo_INFORMATION：テーブル

INFORMATION	PERSONID	氏名	所属	関係	住所	電話番号
1	1	Iiduka	College	honnin	apart	000-0000-0000
2	2	Kawanishi	Graduate	Senior	apart	
3	3	Kitamura	College	colleague	apart	
4	4	Prof.Sato	College	teacher	house	
5	5	nobody	none	none	none	
6	6	Tsai	Graduate	Senior	apart	

4.3.3 「新たな個人を登録する」機能の動作

新規登録画面では、登録画面のデータを元に、PERSON テーブルに新規レコードを付加し、その主キーを、INFORMATION テーブルに新規レコードを付加した際に付加することに成功した。同姓同名の個人を登録した場合警告する機能は機能しなかったものの、要求された機能の大半を実装することに成功した。

（写真 4-10）携帯電話側の動作

（写真 4-11）データベース側の動作（個人情報テーブル）

（写真 4-12）データベース側の動作（顔認識テーブル）

第 5 章　結論

完成には至らなかったものの、顔認識アプリケーションの大半を完成させた。その過程で、携帯電話で機能する WEB アプリケーションをどのように作成するのか、基本的事項とノウハウを獲得する事が出来た。第 1 章でも述べたように携帯電話で機能する WEB アプリケーション技術はユビキタスコンピューティングの大きな流れなので、今回の経験はユビキタスコンピューティングの習得にもつながった。

また、アプリケーションを構築する過程で、WINDOWS と携帯電話双方のプログラミング環境、WEB サービスの実体について理解を深めることが出来た。

第 1 章では今後ユビキタスコンピューティングの主流となりうる携帯電話上で WEB サービスを利用することの大きな可能性を示すことが出来た。

第 2 章では携帯電話上で WEB サービスを利用するのに必要不可欠な要素技術である WINDOWS 上のプログラミング、携帯電話上のプログラミング、画像処理技術機穂について説明した。その結果、携帯電話上の WEB サービスという技術の概略を示すことが出来た。

第 3 章では UML（Unified　Modeling　Language）と呼ばれる設計手法を利用した顔認識アプリケーションの設計を紹介した。この設計手法はオブジェクト指向でプログラミングを中心に現在主流とされるアプリケーションの設計手法であり、今回、標準的なアプリケーションの開発の一例を示した。

第 4 章では携帯電話上で機能する WEB アプリケーションを作成する過程を示し、第 2 章で紹介した技術を実際に応用するに必要なソフトウェア、解決するべき諸問題を紹介した。そして、そうした要素技術を取り込んで、現実の環境に UML を適用してプログラミングを行う内部設計の一例を示した。

今後、今回の顔認識アプリケーションを、図 4-10 の空白部分を埋めた完全なもの、機能するものに仕上げたいと考えている。そのため、以下の項目について、更に研究を続けてゆきたいと考えている。

（1） 顔認識ライブラリを WEB サービスに接続出来るよう改善

（2） 4.1.2.4.2 節で紹介したセッション制御の技術の携帯電話上での応用

（3）個人情報を暗号化して通信するための JAVA 上の暗号技術の利用

謝辞

最後に、論文作成全般にあたって懇切丁寧に指導を頂いた佐藤教授、私の説明に耳を傾け不足な点を指摘してくれた同級生の鎌形君と北村君、顔認識アプリケーションテストのため実験に協力してくれた佐藤研究室・住田研究室の先輩方、論文作成の要所要所で助言をいただいた日向先輩と川西先輩、そして模範となる論文を研究室に残していった井田先輩と玖島先輩そのほかの諸先輩に感謝の意を表明したいと思います。

参考文献

（携帯電話関係）

[1-1] NTTDOCOMO I-mode JAVA 公式ホームページ
http://www.nttdocomo.co.jp/p_s/imode/java/index.html

・NTTDOCOMO i アプリコンテンツ開発ガイド for DoJa3.0 詳細編

・NTTDOCOMO i アプリコンテンツ開発ガイド for DoJa3.0 i アプリオプション拡張編

・DoJa 3.0 for 505iS (オプション API リファレンス)（HTML ファイル）

[1-2] Unofficial “DoCoMo Profile-2.0” （機種別リファレンス）
http://godwood.allnet.ne.jp/vioret/dojaapi2/cpm/nttdocomo/*

[1-3] 「I アプリ プログラミング」（基本チュートリアル）
http://www.geocities.co.jp/SilliconValley-Bay/2972

[1-4] JAVA PROGRAMING for I-application（同じく重要なチュートリアル）
http://www.mdn.co.jp/webcre/Javaprog

[1-5] 「I アプリの作り方（NTTDOCOMO DOJA 編)」
山崎 由喜憲 SOFTBANK Publishing 刊（2001 年 4 月）

[1-6] 「TCP/IP JAVA ネットワーキングプログラミング」
小高 知弘著 オーム社刊（1999 年 8 月）

（SOAP 関係）

[2-1] 連載 WEB サービスのキホン
http://www.atmarkit.co.jp/fxml/tanpatsu/21websvc（WEB サービス基本資料）

[2-2] 連載 ビジネス Web サービス最新事情（セキュリティ関係）
http://www.atmarkit.co.jp/fxml/tanpatsu/25websvc/01.html

[2-3] 連載 SOAP の仕掛け
http://www.atmarkit.co.jp/fxml/rensai/soap01/soap01.html

[2-4] XML/WEB サービス基本
http://www.wakhok.ac.jp/~sakamoto/WS/*

[2-5] 「JAVA による SOAP プログラミング パーフェクトガイド」
技術評論社刊 William Brogden 著 沖林 正紀監訳（2002 年 10 月）

[2-6] 実践！ SVG（SVG 規格（XML 画像規格）に関する参考資料）
http://www.utj.co.jp/xml/dev/svg/*

[2-7] デベロッパーズコーナー： 実践！XSLT
http://www.utj.co.jp/xml/dev/xslt/*（XSLT と DOM に関する参考資料）

[2-8] NanoXML/Java 2.2 マニュアル（携帯 SOAP に関する参考資料）

[2-9]　Apache Axis で Web サービス（KSOAP チュートリアル）
http://members.jcom.home.ne.jp/hitumabushi/wserv/axis.html
[2-10]　XML を知る：ポスト HTML、WEB 上の SGML
リチャード・ライト著　ネティズム工房訳、プレンティース出版刊（1998 年 6 月）
ソフトウェアのダウンロード元として
http://ksoap.enhydra.org/（KSOAP　資料及びライブラリのダウンロード元）

（COM と NET 関係）
[3-1]　MSDN ライブラリ（http://www.microsoft.com/japan/msdn）
チュートリアル：ATL 及び C++マネージ拡張を使用した COM の相互運用性
COM プログラミングの基本（基本原理の説明）
Dr.GUI　コンポーネント、COM,ATL を使う（より詳細な説明）
Microsoft .NET/COM の移行と相互運用性（ASPNET での COM 利用基本資料）
Visual C++での ADO プログラミング（DB 利用法基本）
C++マネージ拡張による COM オブジェクトの NET 環境への公開
ATL のクラスを使ってみる
[3-2]　ヘッダファイルとは（ヘッダファイルに関する詳細な設定法）
http://www.02.246.ne.jp/~torutk/cxx/file/header.html
[3-3]　C++アプリケーションから MSXML を使う（C++環境での XML 利用法（参照））
http://www.utj.co.jp/xml/dev/dom/dxdom*.html
[3-4]　解説　インサイド .NET Framework［改訂版］（NET についての詳細な知識）
http://www.atmarkit.co.jp/fdotnet/technology/idnfw11_index/index.html
[3-5]　「ATL インターナル　－仕組み・設計・実装法－」　アスキー出版局
[3-6]　「日経ソフトウェア別冊　ゼロから始める VISUAL　C++」
日経ソフトウェア編集部編　日経 BP 社刊（2003 年 5 月）
[3-7]　「現場で使える SQL」　小野哲　藤本亮著　翔泳社刊（2001 年 10 月）

（ASPNET と VB 関係）
[4-1]　Visual Basic の基本
http://www.sis.otsuma.ac.jp/~tsutsumi/lecture/graphics/*
[4-2]　VBNET プログラミングリファレンス（VBNET 基礎）
http://www.microsoft.com/japan/msdn/net/vbnetref
[4-3]　Multi Web UI アプリケーション開発への道（ASPNET 基本資料）
http://ww.gotdotnet.com/japan/team/fieldevangelist/ryon/*
[4-4]　DB 設計者のための明解 ADONET（ADONET 基本資料）
http://ww.gotdotnet.com/japan/team/fieldevangelist/fusaito/*

[4-5] ステップアップ ADONET（ADONET の ASP 上での使用法）
http://www.sqlpassj.org/bunkai/web/series/ado
[4-6] 連載　プログラミング ASPNET（ASP 概念説明）
http://www.atmarkit.co.jp/fdotnet/aspnet/aspnet01/aspnet01_02.html
[4-7] Microsoft　ASPNET クイックスタートチュートリアル
http://ja.gotdotnet.com/quickstart/aspplus/doc
[4-8] 「ASP.NET　プログラミング本格入門」
生形洋一　堀田健也著　技術評論社刊（2003 年 10 月）

（OPENCV と画像関係）
[5-1] BMP ファイルから DIB を読み込む（BMP ファイルの取扱法）
http://www.sm.rim.or.jp/~shishiido/readbmp.html
[5-2] CreateDIBSection（BMP ファイルの取扱法 2）
http://www.microsoft.com/japan/developer/library/jpgdipf/
[5-3] VC++-Tips About Library（DLL 取扱法の知識 1）
http://www.bc.wakwak.com/~wmasayoshi/prog/builder/tips/
[5-4] リンク方式について（DLL 取扱法の知識 2）
http://www.microsoft.com/japan/developer/library/vccore/
[5-5] インクチャットを構築する（COM 上での画像取扱法）
http://www.microsoft.com/japan/msdn/tabletpc/techart/binkchat.asp
[5-6] COM 研究室（WINDOWS 上での低レベル画像取扱法）
http://www5.plala.or.jp/atata/com/*
[5-7] Intel Image Processing Library（OPENCV 関係 BMP 操作資料）
http://www.mapletown.net/~nekora/soft/howto/ipl.html
[5-8] 「IPL,OPENCV を用いたプログラミング」
栗田雄一(yuuich-k2is.aist-nara.ac.jp)
ソフトウェアのダウンロード元として
http://www.intel.com（OPENCV 本体、顔認識アプリケーション入手元）
http://www.cvmt.ds/~hn/Images/install/IPL（IPL 本体入手元）
http://www.intel.co.jp（IPL 日本語取扱説明書入手元）

（UML とアプリケーション設計関係）
[6-1] UML モデリングによる設計の流れを理解する（UML の単純明瞭な紹介）
http://www.atmarkit.co.jp/fjava/rensai2/websys03/
[6-2] 初歩の UML（連載シリーズ全 12 回）（UML　教科書）
http://www.atmarkit.co.jp/fjava/devs/renew_uml＊
[6-3] 連載：ここから始めるオブジェクト指向（オブジェクト指向　教科書）
http://www.atmarkit.co.jp/fjava/devs/object*
[6-4] 「かんたん UML」
（株）オージス総研著　翔泳社刊（1999 年 6 月）
[6-5] 「WEB サービスを利用したカスタマイズドウェア製造販売のビジネス構築」
井田　綾香著　筑波大学社会工学類　平成 14 年度卒業論文（2003 年 3 月）
[6-6] 「e-learning による教育支援システム」
玖島　周平著　筑波大学社会工学類　平成 14 年度卒業論文（2003 年 3 月）
[6-7] IT 用語辞典 e-words
http://e-words.jp/?w=.net

（携帯電話と顔認識に関する応用例）
[7-1] 携帯電話料金に関する HP（2003 年 10 月 1 日現在）
http://www.nttdocomo.co.jp/p_s/imode/im/imotion02.html
http://foma.nttdocomo.co.jp/fee/fee_packetpack.html
http://www.au.kddi.com/ezweb/ryokin_course/kanto_chubu.html
[7-2] 携帯電話評価に関する HP（2003 年 10 月 1 日現在）
http://www.au.kddi.com/ezfactory/tec/spec/ezplus.html（EZJAVA）
http://www.au.kddi.com/ezfactory/tec/spec/brew.html（EZBREW）
http://www.nttdocomo.co.jp/p_s/imode/java/index.html（DOJA）
[7-3] 顔認識アルゴリズムとその応用例について（2003 年 10 月 8 日現在）
http://www.cgc.co.jp/htw/npi/060/NEC_021024.html
http://www.njp.co.jp/Gi-ken/1giju2.htm
http://www.face-id.jp/s11-hogo/s11.html
http://www.netscience.ne.jp/NetScience/products/software/ByCategory?ID=126
http://ascii24.com/news/inside/2002/08/08/print/637794.html
http://www.zdnet.co.jp/mobile/0309/11/bioauto.html
http://www.mri.co.jp/SOFTWARE/SOFT/malib.html
[7-4] 指紋認識に関する HP
http://www.fmworld.net/biz/fmv/product/hard/security/fs210/

（その他参考にした資料）

[8-1] 顔認識に関するその他の記事

http://www.hotwired.co.jp/news/news/culture/story/20010810205.html

http://www.hotwired.co.jp/news/news/culture/story/20020108205.html

http://www.wired.com/news/privacy/0,1848,52563,00.html

[8-2] http://www.nicos.co.jp/infrared/about/index.html（携帯電話カード決済）

[8-3] http://www.atmarkit.co.jp/fmobile/rensai/hdml04/hdml04.html#4

（携帯電話の環境変数に関する資料）

[8-4] **「研究室選びのための掲示板について」**

平成 13 年度経営工学専攻　**総合研究発表　飯塚　岳郎 (2003 年 2 月)**

付録

付録として、今回、顔認識アプリケーションを作成するのに用いた以下のソースコードを添付する。ただし、ここに添付したのは利用したソースコードのうち自作の部分、あるいは入手したプログラムのうち徹底的な改造を施した部分のみである。

（付録 1）FaceLibrary2（顔認識ライブラリ）

1.1	FaceLibrary3.cpp/.h	NET 形式のプログラムへのインターフェース
1.2	FaseBase..cpp/.h	顔認識ライブラリ本体
1.3	ContEHMM.cpp/.h	パターン認識情報の管理・入出力を担当
1.4	StdAfx2.h	標準インクルードファイル（COM コンポーネント用）
1.5	Stdafx.h	標準インクルードファイル（NET プログラム用）

（付録 2）Webservice1（WEB サービス本体）

（付録 3）KSOAP シリーズ（携帯電話上のクライアント）

3.1	KSOAP1.java	ユースケース「個人の認識」を担当する GUI
3.2	KSOAP2.java	ユースケース「個人情報の検索、修正」を担当する GUI
3.3	KSOAP3.java	ユースケース「新規個人の登録」を担当する GUI
3.4	HttpTransport.java	KSOAP クライアントで通信部分を担当 （改造を相当施したため掲載）

（注）.cpp/.h とある部分があるが、これは C++ソースファイル（CPP）とインクルードファイル（h）が 2 つで一組となってプログラムを構成することを意味する。

```
（付録 1）FaceLibrary2（顔認識ライブラリ）
////////////////////////////////////////////////////////////////////////////
//
//　このプログラムは Intel License Agreement に基づく Intel 社のオープンソースのプログラムである。
//　私、飯塚岳郎が卒業研究の目的でいくつかの重要な改変を行った。
//　　（１）ATL に基づく COM コンポーネントへの改装
//　　（２）ファイルアクセスのデータベース化
//　　（３）IPL 導入による BMP<->IPL（OPENCV）間のフォーマット変換機能の導入
//
//　このプログラムも Intel License Agreement に準じて公開する。
//　なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負うものとする
//
//　（本ライブラリに必要な DLL ファイル）
//　highgui.dll, cv.dll, cvaux.dll(OPENCV)
//　IPL.dll, IPLa6.dll(IPL)
//
//　（本ファイルの内容）FaceLibrary3.h
//　顔認識ライブラリの NET 用インターフェースのインターフェース
//　（MgdFindFaceLibrary クラスのインターフェース）
//　（従って、INTEL 社のプログラムとは直接の関連はない）
//
////////////////////////////////////////////////////////////////////////////*/

#pragma once

/* プログラムに必要なインクルードファイル */
#include "stdafx2.h"
#include "facebase.h"

/* 名前空間の宣言（NET プログラミングにおける実質上のインクルードファイルの宣言） */
using namespace System;
using namespace System::Runtime::InteropServices;
using namespace FACEBASE;
```

```
38
39  namespace FaceLibrary3
40  {
41          public __gc class MgdFindFaceLibrary
42          {
43                  private:
44                          /* 作業に必要なインスタンス */
45                          FACEBASE::CFaceBase *pDataBase;// 顔認識エン
46  ジン（CFaceBase）へのポインタ
47                          System::String *errorTextFace;        // エラー文字
48  列
49
50                  public:
51
52                          /* 顔認識エンジンへのインターフェース */
53                          MgdFindFaceLibrary()
54                          {
55                                  pDataBase = new FACEBASE::CFaceBase();
56                          }
57                          BOOL Load(void);
58                          BOOL Save(void);
59                          BOOL Unload(void);
60                          BOOL Load_Config(void);
61                          BOOL Save_Config(void);
62                          BOOL      SetDBInfo(System::String**      Provider,
63  System::String** Source, System::String** DB_Name, System::String**
64  UserName, System::String** PassWord);
65                          BOOL      SetTABLEInfo(System::String**      info,
66  System::String** image, System::String** config);
67                          BOOL errorReport(System::String** errorText);
68                          BOOL      AddImage(System::Int64      PersonID,
69  System::Byte imageData[], BOOL loaded);
70                          BOOL RecognizePerson(System::Byte imageData[],
71  System::String** result, System::String** probresult);
72                          BOOL      RecognizePerson_Checked(System::Byte
73  imageData[], System::String** remove_key, System::String** result,
74  System::String** probresult);
```

```
75                          BOOL RecognizePerson2(System::Byte imageData[],
76  System::Int64 result, System::Single probresult);
77                          BOOL TrainPerson(System::Int64 PersonID);
78                          BOOL TrainAll(void);
79                  private:
80
81                          /* インターフェース内部で処理に必要なメソッド */
82                          HRESULT LoadCvvImage( System::Byte
83  imageData[], CvvImage *image, CRect *rect );
84
85          };
86  }
87
88  ////////////////////////////////////////////////////////////////////////////
89  //
90  //  このプログラムは Intel License Agreement に基づく Intel 社のオープンソ
91  ースのプログラムである。
92  //  私、飯塚岳郎が卒業研究の目的でいくつかの重要な改変を行った。
93  //    （1）ATL に基づく COM コンポーネントへの改装
94  //    （2）ファイルアクセスのデータベース化
95  //    （3）IPL 導入による BMP<->IPL（OPENCV）間のフォーマット変換機
96  能の導入
97  //
98  //  このプログラムも Intel License Agreement に準じて公開する。
99  //  なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負
100 うものとする
101 //
102 //  （本ライブラリに必要な DLL ファイル）
103 //  highgui.dll, cv.dll, cvaux.dll(OPENCV)
104 //  IPL.dll, IPLa6.dll(IPL)
105 //
106 //  （本ファイルの内容）FaceLibrary3.h
107 //  顔認識ライブラリの NET 用インターフェースの実装
108 （MgdFindFaceLibrary クラスの実装）
109 //  従って、INTEL 社のプログラムとは直接の関連はない
110 //
111 ////////////////////////////////////////////////////////////////////////////*/
```

```
112
113 /* プログラムに必要なインクルードファイル */
114 #include "stdafx.h"
115 #include "FaceLibrary3.h"
116
117 namespace FaceLibrary3
118 {
119
120         ////////////////////////////////////////////////////////////////
121         // 顔認識テーブル、画像テーブルへのアクセス
122         ////////////////////////////////////////////////////////////////
123         /* 顔認識情報の読込メソッドへのインターフェース
124             ユースケース記述では「個人の認識」部分で、顔認識を行う前に読
125 込 */
126         BOOL MgdFindFaceLibrary::Load(void)
127         {
128                 return pDataBase->Load();
129                 return true;
130         }
131
132         /* 顔認識情報の保存メソッドへのインターフェース
133             顔認識、パターン学習の成果をこのメソッドを利用してデータベー
134 スに保存   */
135         BOOL MgdFindFaceLibrary::Save(void)
136         {
137                 return pDataBase->Save();
138                 return true;
139         }
140
141         /* 顔認識エンジンの終了処理に対するインターフェース
142             ユースケース記述の「個人情報の検索」部分の最後に用いる */
143         BOOL MgdFindFaceLibrary::Unload(void)
144         {
145                 pDataBase->Unload();
146                 return S_OK;
147         }
148
```

```
149         /* 顔認識エンジンの設定情報を取り扱うインターフェース　*/
150         /* 顔認識エンジンを利用するには絶対不可欠な処理であり、二番目に
151 行う処理である　*/
152         BOOL MgdFindFaceLibrary::Load_Config(void)
153         {
154                 return pDataBase->load_config();
155                 return true;
156         }
157
158         BOOL MgdFindFaceLibrary::Save_Config(void)
159         {
160                 return pDataBase->save_config();
161                 return true;
162         }
163
164         /* 顔認識エンジンのデータベース接続先を読み込む　*/
165         /* 顔認識エンジンを利用するには絶対不可欠な処理であり、一番最初
166 に行う処理である　*/
167         BOOL MgdFindFaceLibrary::SetDBInfo(System::String** Provider,
168 System::String** Source, System::String** DB_Name, System::String**
169 UserName, System::String** PassWord)
170         {
171                 // TODO: ここに実装コードを追加してください。
172                 System::String __pin *Provider1 = *Provider;
173                 System::String __pin *Source1 = *Source;
174                 System::String __pin *DB_Name1 = *DB_Name;
175                 System::String __pin *UserName1 = *UserName;
176                 System::String __pin *PassWord1 = *PassWord;
177
178                 HRESULT                 info                    =
179 pDataBase->SetDBInfo( CString(Provider1),
180
181         CString(Source1),
182
183         CString(DB_Name1),
184
185         CString(UserName1),
```

```
186
187             CString(PassWord1) );
188                     return true;
189             }
190
191             BOOL  MgdFindFaceLibrary::SetTABLEInfo(System::String**  info,
192     System::String** image, System::String** config)
193             {
194                     System::String __pin *info1 = *info;
195                     System::String __pin *image1 = *image;
196                     System::String __pin *config1 = *config;
197
198             pDataBase->SetTABLEInfo( CString(info1),CString(image1),CStrin
199     g(config1) );
200                     return true;
201             }
202
203             /* 顔認識エンジンの作動状況を取得するためのインターフェース  */
204             BOOL            MgdFindFaceLibrary::errorReport(System::String**
205     errorText)
206             {
207                     CString errorTextEdit = pDataBase->reportError();
208                     //errorTextEdit += errorTextFace;
209                     if( errorTextEdit.IsEmpty() )
210                             errorTextEdit = "All is fine.";
211                     System::String          __pin          *errorText1          =
212     errorTextEdit.AllocSysString();
213                     *errorText = errorText1->ToString();
214                     *errorText = (*errorText)->Insert( 0, errorTextEdit );
215                     return true;
216             }
217
218             ////////////////////////////////////////////////////////////////////
219             // 顔認識エンジンの実際の業務へのインターフェース
220             ////////////////////////////////////////////////////////////////////
221             /* 顔認識エンジンの保有するデータに対して顔写真データを付加  */
222             /* ユースケース記述「個人情報の検索」に相当する作業で利用  */
```

```
223         BOOL   MgdFindFaceLibrary::AddImage(System::Int64   PersonID,
224 System::Byte imageData[], BOOL loaded)
225         {
226                 // TRUE なら読込済み、FALSE ならこれから読込
227
228                 // データの受け取り
229                 using namespace FACEBASE;
230                 HRESULT hr = S_OK;
231                 CRect    rect;
232                 CString          errorTextHere;
233
234                 // ポインタの処理
235                 errorTextHere.AppendFormat(_T("LOADED    =    %d¥n"),
236 loaded );
237                 if( loaded != 0 )
238                 {
239                         pDataBase->AddImage( PersonID, NULL, true );
240                         return true;
241                 }
242
243                 // 画像の処理
244                 CvvImage *image = new CvvImage();
245                 hr = LoadCvvImage( imageData, image, &rect );
246                 if( FAILED(hr))
247                         return false;
248
249                 // 関数の呼び出し
250                 pDataBase->AddImage( PersonID, image, false );
251
252                 // エラー報告
253                 errorTextFace        =        errorTextFace->Insert(        0,
254 errorTextHere.AllocSysString() );
255                 return true;
256         }
257
258         /* 顔認識エンジンの認識メソッドへのインターフェース
259             ユースケース記述「個人の識別」で利用   */
```

```
260         /* 最も基本的なインターフェース、該当者リストの作成に利用 */
261         BOOL        MgdFindFaceLibrary::RecognizePerson(System::Byte
262 imageData[], System::String** result, System::String** probresult)
263         {
264                 // 必要な作業用変数の宣言
265                 using namespace FACEBASE;
266
267                 long key[REGNZ_NUMBER+1];
268                 float likelihood[REGNZ_NUMBER+1];
269                 CvvImage *image = new CvvImage();                   // 画像
270 関係
271                 CRect    *rect = new CRect();
272                 HRESULT hr;
273                 // エラー報告用変数
274
275                 // 画像の読込
276                 hr = LoadCvvImage( imageData, image, rect );
277                 if( FAILED(hr))
278                         return false;
279
280                 // 関数の呼び出し
281                 pDataBase->RecognizePerson( image, CRect(0,0,0,0), key,
282 NULL, likelihood );
283
284                 // 結果の書き出し(問題点も多いがあえて現実的であるために)
285                 CString text1,text2;
286                 for( int i = 0; i < REGNZ_NUMBER; i++ )
287                 {
288                         text1.AppendFormat(_T("%d,"), key[i] );
289                         text2.AppendFormat(_T("%f,"), likelihood[i] );
290                 }
291
292                 *result = text1.AllocSysString();
293                 *probresult = text2.AllocSysString();
294                 return true;
295         }
296
```

```
297         /* 除外リストを考慮に入れたインターフェース、該当者リストのスク
298 ロール処理を考慮 */
299         BOOL
300 MgdFindFaceLibrary::RecognizePerson_Checked(System::Byte
301 imageData[], System::String** remove_key, System::String** result,
302 System::String** probresult)
303         {
304                 // 必要な作業用変数の宣言
305                 using namespace FACEBASE;
306
307                 long key[REGNZ_NUMBER+1];
308                 float likelihood[REGNZ_NUMBER+1];
309                 CvvImage *image = new CvvImage();                    // 画像
310 関係
311                 CRect    *rect = new CRect();
312                 HRESULT hr;
313                 // エラー報告用変数
314
315                 // 画像の読込
316                 hr = LoadCvvImage( imageData, image, rect );
317                 if( FAILED(hr))
318                         return false;
319
320                 // 結果キーの解読(警告：これはかなり問題のあるコード)
321                 System::String __pin *remove_key1 = *remove_key;
322                 CString removeString = CString( remove_key1 );
323                 // 必要な変数
324                 CString resToken;
325                 int curPos = 0;
326                 long remove_key_data[100];
327                 for( int i = 0; i < 100; i++ )
328                         // 変数の初期化
329                         remove_key_data[i] = 0;
330                 if( !removeString.IsEmpty() )
331                 {
332                         for( int i = 0; curPos < removeString.GetLength();
333 i++ )
```

```
334                         {
335                                 resToken = removeString.Tokenize( " ",
336 curPos );
337                                 sscanf( resToken, "%ld", ( remove_key_data +
338 i ) );
339                         }
340                 }
341
342                 // 顔認識の実行
343                 if( !removeString.IsEmpty() )
344                         pDataBase->RecognizePerson( image, CRect(0,0,0,0),
345 key, remove_key_data, likelihood );
346                 else
347                         pDataBase->RecognizePerson( image, CRect(0,0,0,0),
348 key, NULL, likelihood );
349
350                 // 結果の書き出し(問題点も多いがあえて現実的であるために)
351                 CString text1, text2;
352                 for( int i = 0; i < REGNZ_NUMBER; i++ )
353                 {
354                         text1.AppendFormat(_T("%d,"), key[i] );
355                         text2.AppendFormat(_T("%f,"), likelihood[i] );
356                 }
357
358                 *result = text1.AllocSysString();
359                 *probresult = text2.AllocSysString();
360                 return true;
361         }
362
363         /* 配列を配列によって入出力するインターフェース、ただし機能せず
364 */
365         BOOL        MgdFindFaceLibrary::RecognizePerson2(System::Byte
366 imageData[], System::Int64 result, System::Single probresult)
367         {
368                 // TODO: ここに実装コードを追加してください。
369                 // 必要な作業用変数の宣言
370                 using namespace FACEBASE;
```

```
371
372                 long key[REGNZ_NUMBER+1];
373                 float likelihood[REGNZ_NUMBER+1];
374                 CvvImage *image = new CvvImage();                   // 画像
375 関係
376                 CRect    *rect = new CRect();
377                 HRESULT hr;
378                 // エラー報告用変数
379
380                 // 画像の読込
381                 hr = LoadCvvImage( imageData, image, rect );
382                 if( FAILED(hr))
383                         return false;
384
385                 // 関数の呼び出し
386                 pDataBase->RecognizePerson( image, CRect(0,0,0,0), key,
387 NULL, likelihood );
388
389                 // 結果の書き出し(問題点も多いがあえて現実的であるために)
390                 //Marshal::Copy( key, result, 0, REGNZ_NUMBER+1 );
391                 //Marshal::Copy( likelihood, probresult, 0,
392 REGNZ_NUMBER+1 );
393                 return true;
394         }
395
396         /* 指定された人間（KEY）に対して顔認識トレーニングを実施
397             ユースケース記述「個人情報の検索」で行う顔認識のインターフェ
398 ース */
399         BOOL MgdFindFaceLibrary::TrainPerson(System::Int64 PersonID)
400         {
401                 pDataBase->TrainPerson( PersonID );
402                 return S_OK;
403         }
404
405         BOOL MgdFindFaceLibrary::TrainAll(void)
406         {
407                 pDataBase->TrainAll();
```

```
408                 return S_OK;
409         }
410
411         /* WEB サービス、または外部のプログラムから受け取ったデータを
412 OPENCV 形式の画像に変換
413             ユースケース記述「個人の認識」で受信した配列データはここで画
414 像データに変換 */
415         HRESULT    MgdFindFaceLibrary::LoadCvvImage(    System::Byte
416 imageData[], CvvImage *image, CRect *rect )
417         {
418                 // データの受け取り
419                 using namespace FACEBASE;
420                 HRESULT hr;
421                 CString          errorTextHere;
422
423                 // 外部の関数による__GC 形式の配列を SAFEARRAY 配列に
424 変換する
425                 //データを格納する BYTE 型の SafeArray 配列の宣言
426 77878(TARGET) 77864(NOW)
427                 SAFEARRAY FAR *psa;
428                 SAFEARRAYBOUND rgsabound[1];
429                 rgsabound[0].lLbound = 0;
430                 rgsabound[0].cElements = imageData->get_Length();
431                 psa = SafeArrayCreate(VT_UI1,1,rgsabound);
432                 if( psa == NULL )
433                         return false;
434
435                 // バイト型のポインタと配列を結びつける（ATTACH する）
436                 byte *pb;
437                 hr = SafeArrayAccessData( psa, ( void ** )&pb );
438                 if( FAILED( hr ) )
439                         return NULL;
440                 else if( !pb )
441                         return NULL;
442
443                 //変換した結果の配列への書き込み
444                 Marshal::Copy(imageData,          0,          pb,
```

```
445 imageData->get_Length());
446                     ::SafeArrayUnaccessData( psa );
447
448                     // 外部の関数より受け取った配列より CVV 形式のデータを読
449 み込み
450                     hr = SafeArrayToCvvImage( psa, image );
451                     if (FAILED(hr))
452                     {
453                             errorTextHere.AppendFormat(_T("TROUBLES:
454 CreateStreamFromSafeArray in DLL¥n") );
455                             return E_FAIL;
456                     }
457
458                     // ROI オプションに関する数値の設定
459                     rect->SetRect( 0, 0, image->Height(), image->Width() );
460
461                     // エラー報告
462                     errorTextFace = errorTextHere.AllocSysString();
463                     return S_OK;
464             }
465 }
466
467 /*M///////////////////////////////////////////////////////////////////////////////////
468 //
469 //     IMPORTANT: READ BEFORE DOWNLOADING, COPYING,
470 INSTALLING OR USING.
471 //
472 //   By downloading, copying, installing or using the software you agree to
473 this license.
474 //   If you do not agree to this license, do not download, install,
475 //   copy or use the software.
476 //
477 //
478 //                                 Intel License Agreement
479 //                         For Open Source Computer Vision Library
480 //
481 // Copyright (C) 2000, Intel Corporation, all rights reserved.
```

```
// Third party copyrights are property of their respective owners.
//
// Redistribution and use in source and binary forms, with or without modification,
// are permitted provided that the following conditions are met:
//
//    * Redistribution's of source code must retain the above copyright notice,
//      this list of conditions and the following disclaimer.
//
//    * Redistribution's in binary form must reproduce the above copyright notice,
//      this list of conditions and the following disclaimer in the documentation
//      and/or other materials provided with the distribution.
//
//    * The name of Intel Corporation may not be used to endorse or promote products
//      derived from this software without specific prior written permission.
//
// This software is provided by the copyright holders and contributors "as is" and
// any express or implied warranties, including, but not limited to, the implied
// warranties of merchantability and fitness for a particular purpose are disclaimed.
// In no event shall the Intel Corporation or contributors be liable for any direct,
// indirect, incidental, special, exemplary, or consequential damages
// (including, but not limited to, procurement of substitute goods or services;
// loss of use, data, or profits; or business interruption) however caused
// and on any theory of liability, whether in contract, strict liability,
// or tort (including negligence or otherwise) arising in any way out of
// the use of this software, even if advised of the possibility of such damage.
//
//   （補足条項）
//   このプログラムは Intel License Agreement に基づく Intel 社のオープンソースのプログラムである。
```

```
519  //   私、飯塚岳郎が卒業研究の目的でいくつかの重要な改変を行った。
520  //     （１）ATL に基づく COM コンポーネントへの改装
521  //     （２）ファイルアクセスのデータベース化
522  //     （３）IPL 導入による BMP<->IPL（OPENCV）間のフォーマット変換機
523  能の導入
524  //
525  //   このプログラムも Intel License Agreement に準じて公開する。
526  //   なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負
527  うものとする
528  //
529  //   （本ファイルの内容）facebase.h
530  //   顔認識ライブラリのヘッダファイル
531  //   （CFaseBase クラス, CPerson クラス, CPersonImg クラスのインターフェ
532  ース）
533  //
534  //   （本ライブラリに必要な DLL ファイル）
535  //   highgui.dll, cv.dll, cvaux.dll(OPENCV)
536  //   IPL.dll, IPLa6.dll(IPL)
537  //
538  ////////////////////////////////////////////////////////////////////////*/
539
540  /* インクルードガード */
541  #ifndef INCLUDE_FACEBASE
542  #define INCLUDE_FACEBASE
543
544  #if  !defined(AFX_FACEBASE_H__76CC3F61_D0C7_4CF8_8479_84950282
545  D632__INCLUDED_)
546  #define
547  AFX_FACEBASE_H__76CC3F61_D0C7_4CF8_8479_84950282D632__INC
548  LUDED_
549
550  #if _MSC_VER > 1000
551  #pragma once
552  #endif // _MSC_VER > 1000
553
554  //   ContEHMM 系統のヘッダファイル
555  #include "stdafx2.h"
```

```
#include "ContEHMM.h"
#define TRAIN_ALL 1
#define TRAIN_UNTRAINED 2

/* 顔情報エンジンの名前空間 FACEBASE の宣言 */
namespace FACEBASE
{
	//クラスの事前宣言（プログラムの解析に必要な）
	class CFaceBase;		//顔データベース本体
	class CBaseInfo;		//データベースの基本情報を格納
	class CPerson;		//個人情報を格納
	class CPersonTrait;		//個人情報リストの操作用メソッドの
定義
	class CPersonImage;		//顔写真情報を格納
	class  CPersonImageTrait;//顔写真情報リストの操作用メソッドの定
義

	//リスト型データ構造の宣言
	typedef CAtlList<CPersonImage*> CPersonImgList;
	typedef CAtlList<CPerson*> CPersonList;

	/* データベースの基本情報を取り扱うクラス */
	class CBaseInfo
	{

	public:
		// 基本情報の設定
		HRESULT SetDBInfo( const CString& PROVIDER,
					const CString& SOURCE,
					const CString& DB,
					const CString& USER,
					const CString& PASSWD );
		HRESULT SetTABLEInfo( const CString& info, const
CString& image, const CString& config );

		// 必要な出力情報の作成
		CString makeup_destination();		// データベース接続を
```

```
作成するのに必要な接続先文を作成
                HRESULT openDB();                       // データベース接続を作成
                CString BaseName(){ return m_infoBase;}
                CString ImageBaseName(){ return m_imageBase;}
                CString ConfigBaseName(){ return m_configBase;}
                _ConnectionPtr get_ConnectionPtr(){ return m_pConnection;}

        protected:
                // 情報そのもの
                CString m_Provider;             // ODBC 接続先サービス（SQLSERVER、MYSQL など）
                CString m_DataSource;           // 相手側コンピュータ名（リモートであればアドレスも）
                CString m_dbName;               // データベース名
                CString m_userName;             // ユーザー名
                CString m_passWord;             // パスワード
                CString m_infoBase;             // 基本情報格納先
                CString m_imageBase;            // 画像情報格納先
                CString m_configBase;           // 設定情報格納先

                // 情報そのもの
                _ConnectionPtr m_pConnection;
        };

        //顔データベース本体
        class CFaceBase
        {

        public:
                /* コンストラクタとデストラクタ */
                CFaceBase();
                virtual ~CFaceBase();

                /* 顔情報 DB のデータベース関係の設定と操作 */
                HRESULT SetDBInfo( const CString& PROVIDER,
```

```
630         // DB に関与する変数の読み取り
631                                                 const CString& SOURCE,
632                                                 const CString& DB,
633                                                 const CString& USER,
634                                                 const CString& PASSWD );
635                 HRESULT SetTABLEInfo( const CString& info,
636                                                         const CString& image,
637                                                         const           CString&
638 config);
639                 CString GetInfoBase(){ return baseInfo.BaseName();}
640                 CString                 GetImageBase(){                 return
641 baseInfo.ImageBaseName();}
642
643                 /* 顔情報データベースのパラメータ設定 */
644                 int   SetParams( //サンプリングに関するパラメータ
645                         CvSize dctSize,
646                         CvSize obsSize,
647                         CvSize delta,
648                         //HMM params
649                         int* states, int mix,
650                         //利用する画像に関するパラメータ
651                         BOOL use_width, int width,
652                         BOOL use_height, int height,
653                         BOOL suppress_intens,
654                         BOOL leave_hmm_alive = FALSE );
655
656                 /* 顔情報データベースの入出力操作 */
657                 HRESULT Load();
658                                         /* 顔情報ファイルの読込 */
659                 void    Unload();
660                                 /* 顔認識ベースのクリア */
661                 HRESULT Save();
662                                         /* 顔情報ファイルの保存 */
663                 HRESULT load_config();
664                                 /* 顔情報ファイルのパラメータ読込 */
665                 HRESULT save_config();
666                                 /* 顔情報ファイルのパラメータ保存 */
```

```
667
668                 /* 顔情報データベースに対するデータ操作 */
669                 long    AddPerson();
670                                                 /* データベースへの個人名追
671 加 */
672                 void     RemovePerson( const long& PersonID );
673                                 /* データベースからの個人名削除 */
674                 CPersonList& GetPersonList() { return m_base; };
675                         /* 登録済み個人名リストの入手 */
676
677                 void    AddImage( long personID, CvvImage* image, bool
678 loaded );       /* 画像関係の入出力 */
679                 void    RemoveImage( long personID, POSITION pos );
680
681                 /* 顔情報データベースに対するデータベース検索 */
682                 CPerson* GetPerson( const int& index );
683                 /* 顔情報ベース内部の個人情報の入手（インデックスより） */
684                 CPerson* FindPersonByName( const CString& name );
685         /* 顔情報ベース内部の個人情報の入手（名前より） */
686                 CPerson* FindPersonByPersonID( const long& PersonID );
687                 int         GetPersonIndex( CPerson* person );
688                 /* 顔情報ベース内部の個人情報インデックスの入手 */
689
690                 /* 顔認識関数（本体） */
691                 int                 RecognizePerson( CvvImage *image, CRect
692 rect, long *key, long *remove_key, float *likelihood );
693
694                 /* 顔認識のトレーニング関数 */
695                 void    TrainAll();
696                                 /* 全顔情報に対する認識トレーニング実行 */
697                 void    TrainPerson( long PersonID );
698         /* 指定された顔情報に対する認識トレーニング実行 */
699
700                 /* 顔情報データベースに対する認識関係の操作 */
701                 int       GetTrainedIndex() { return m_trained_index; }
702                         /* 顔情報ベースより現在認識トレーニング中の個人情
703 報インデックス確保 */
```

```
704                 void      SetTrainedIndex( int index ) { m_trained_index =
705 index; }/* 顔情報ベースに現在認識トレーニング中の個人情報インデックス設
706 定 */
707                 void      DeleteHMMInfo();
708                                                   /* 顔情報ベースの設定情報削
709 除 */
710                 void      SetModified( bool modified = true ) { m_modified =
711 modified; }     /* 最終画像が更新済であることの登録 */
712                 bool      IsModified() { return m_modified; }
713                           /* 最終画像が更新済であることの確認 */
714
715                 /* 顔情報データベースに対する画像関係の操作 */
716                 CvvImage *GetTrainedImage() { return m_trained_image; }
717                       /* 顔情報ベース内部の認識トレーニング済顔画像の入
718 手 */
719                 void       SetImageSize( CSize size );
720                           /* 画像サイズの設定 */
721                 CSize      GetImageSize() { return m_baseImgSize; };
722                       /* 画像サイズの読込 */
723
724                 /* 顔情報データベースのユーティリティ */
725                 CString   reportError();
726
727           /* 顔情報データベースの基本情報 */
728           public:
729                 // 顔情報認識のパラメータ
730                 int               m_stnum[32];                    // 顔 認 識
731 （HMM)に関するパラメータ
732                 int               m_mixnum[128];
733
734                 CvSize m_delta;                                   //サンプリング
735 に関するパラメータ
736                 CvSize m_obsSize;
737                 CvSize m_dctSize;
738                 BOOL  m_suppress_intensity;//圧縮の有無の登録
739
740                 // 顔情報認識のパラメータ
```

```
741                 BOOL  m_useWidth;                       //利用する画像
742 に関するパラメータ
743                 BOOL  m_useHeight;
744                 int             m_scaleWidth;
745                 int             m_scaleHeight;
746
747         protected:
748                 CBaseInfo baseInfo;                     /* 顔情報データベース
749 の本質的情報 */
750                 CPersonList m_base;                     /* 個人情報リストへの
751 ポインタ */
752                 CSize     m_baseImgSize;                /* 画像サイズ情報 */
753                 bool      m_modified;
754                 int       m_trained_index;
755                 CvvImage *m_trained_image;/* 現在認識関数が処理中の画
756 像 */
757                 bool      image_load;
758
759         };
760
761         /* 個人情報ファイルのクラス */
762         class CPerson
763         {
764
765         public:
766                 /* コンストラクタとデストラクタ */
767                 CPerson( CFaceBase* parent );
768                 virtual ~CPerson();
769
770                 /* 個人情報の登録と削除 */
771                 void      SetName( const CString& name );       /* 個人
772 名の操作 */                                                    /* 個人名の登
773 録 */
774                 void      SetPersonID( long PersonID );
775
776                 const CString& GetName() { return m_name; }
777                 long      GetPersonID(){ return m_PersonID; }
```

```
778
779                 /* 個人情報の入出力操作 */
780                 HRESULT   Load( _RecordsetPtr m_pRecordSet );   /* 個人
781 情報の読込、保存、クリア */
782                 void      Unload();
783                 HRESULT   Save( _RecordsetPtr m_pRecordSet );
784                 CPersonImgList& GetImgList() { return m_imgs; }   /* 画像
785 ファイルへのアクセスを確保 */
786         /* 画像ファイルへのアクセスを確保 */
787                 HRESULT      ImageDownLoader(          _RecordsetPtr
788 m_pRecordSet, LPCSTR dataROW, CvvImage *newImage );
789                 HRESULT   ImageUpLoader( _RecordsetPtr m_pRecordSet,
790 LPCSTR datarow, CvvImage *newImage );
791
792                 /* 画像情報の入出力操作（直接実装） */
793                 void      AddImage( CvvImage* import_image, CRect rect );
794         /* 画像ファイルの登録と削除 */
795                 void      RemoveImage( POSITION pos );
796                 void      MoveToTop( POSITION pos );
797
798                 /* 個人情報の設定と操作 */
799                 void      SetModified( bool modified = true )          /* 最終
800 画像の更新の有無の設定と確認 */
801                 {
802                     m_modified = modified;
803                     if( modified )
804                             m_trained = false;
805                 }
806                 bool                IsModified() { return m_modified; }
807                 CFaceBase*          GetParentBase()      {      return
808 m_parent; }/* アクセス元である顔情報データベースの報告 */
809
810                 /* HMM 学習情報の設定と操作 */
811                 void                TrainHMM();
812                         /* 顔認識トレーニングの実施 */
813                 bool                IsTrained()  {  return  m_trained;  };
814         /* 認識トレーニングの有無の確認 */
```

```
815                 void                DeleteHMMInfo();
816                 void                ClearHMM();
817                 CContEHMM&              GetHMM();
818
819         /* 個人情報ファイルの本質的情報 */
820         public:
821                 bool                m_trained;      /* 個人情報に対する重
822 要なオプション */
823                 long                m_PersonID;   /* 個人情報の主キー
824 */
825                 CString                 m_name;             /* 登録
826 された個人名 */
827                 CContEHMM           *m_hmm;             /* HMM 学習結
828 果 */
829                 CPersonImgList      m_imgs;             /* メモリ上の
830 画像情報へのアクセス（リストのポインタ） */
831         protected:
832                 CFaceBase*          m_parent;       /* 登録元である顔情報
833 データベースへのポインタ */
834                 bool                m_modified;
835         };
836
837         /* 個人画像を取り扱うクラス */
838         class CPersonImage
839         {
840
841         public:
842                 /* コンストラクタとデストラクタ */
843                 CPersonImage();
844                 virtual ~CPersonImage();
845
846                 /* 画像ファイルの読込と操作 */
847                 void      Unload();
848                 CvvImage&     GetImage() { return m_img; }
849                 void     SetImage( CvvImage image ) { m_img = image; };
850                 void      SetRoiInFile( CRect r );
851                 CRect     GetRoiInFile() { return m_roi_in_file; }
```

```
852
853                 /* 主要な画像クラスに対する操作 */
854                 void    Set_ImageID( long ImageID ){ m_ImageID =
855 ImageID; }
856                 long    Get_ImageID(){ return m_ImageID; }
857                 void    SetModified( bool modified = true ) { m_modified =
858 modified; }
859                 bool    IsModified() { return m_modified; }
860
861         protected:
862                 /* コンストラクタとデストラクタ */
863                 long      m_ImageID;
864                 CvvImage m_img;
865                 CRect      m_roi_in_file;
866                 bool       m_modified;
867         };
868
869         //グローバル関数（エラー処理系統その他）
870         inline void TESTHR( HRESULT _hr );
871         HRESULT errorSQLInside( _com_error &e );
872         void errorSQLProvider( _ConnectionPtr m_pConnection );
873         // 配列型データに対して入出力ストリームを確立する
874         HRESULT        CreateStreamFromSafeArray(        SAFEARRAY
875 *pvSafeArray, IStream **pplStream );
876         // 配列型データに対してメモリ領域を確保
877         HRESULT CopySafeArrayToHMem( SAFEARRAY *pvSafeArray,
878 HANDLE *hMem );
879         // 配列型データを CVVIMAGE 型データに変換
880         HRESULT SafeArrayToCvvImage( SAFEARRAY *pvSafeArray,
881 CvvImage *newImage );
882         // 配列型データに CVVIMAGE 型データを変換
883         SAFEARRAY *SafeArrayFromCvvImage( CvvImage *newImage );
884
885         //グローバル関数（その他：元からの関数群）
886         void LightingCorrection(IplImage* ipl_image);
887         void NormalizeIntensity(IplImage* ipl_image, int level);
888         void NormalizeImageForHMM( IplImage* ipl_scaled, IplImage*
```

```
889 normalized_image );
890         void ExtractDCT( float* src, float* dst, int num_vec, int dst_len );
891
892 }
893 #endif
894 // !defined(AFX_FACEBASE_H__76CC3F61_D0C7_4CF8_8479_84950282D
895 632__INCLUDED_)
896
897 // 基本エラー情報の設定
898 #define MEMORYSHORT "メモリ不足で割り当てられません。"
899 #define FILENOTREAD "ファイルを読み込めません。"
900 #define FILENOTOPEN "ファイルを開けません。"
901 #define TROUBLES   "問題があり完了できませんでした。"
902 #define ATTACHFAIL "割り当て失敗"
903 #define ISVACANT    "空です"
904
905 // 基本的な定数の設定
906 #define CHUNKSIZE  10000  //レコード処理用のバッファサイズ
907 #define BUFSIZE            200000 //画像格納用バッファサイズ
908 #define REGNZ_NUMBER 3          //最大候補数
909 #define LIMIT_RECOGNIZE 10      //最大トレーニング用画像数（トレーニ
910 ングの効率性より）
911
912
913
914 #endif
915
916 ////////////////////////////////////////////////////////////////////////////
917 //
918 //    IMPORTANT: READ BEFORE DOWNLOADING, COPYING,
919 INSTALLING OR USING.
920 //
921 //  By downloading, copying, installing or using the software you agree to
922 this license.
923 //  If you do not agree to this license, do not download, install,
924 //  copy or use the software.
925 //
```

```
926 //
927 //                          Intel License Agreement
928 //                  For Open Source Computer Vision Library
929 //
930 // Copyright (C) 2000, Intel Corporation, all rights reserved.
931 // Third party copyrights are property of their respective owners.
932 //
933 // Redistribution and use in source and binary forms, with or without
934 modification,
935 // are permitted provided that the following conditions are met:
936 //
937 //   * Redistribution's of source code must retain the above copyright notice,
938 //     this list of conditions and the following disclaimer.
939 //
940 //   * Redistribution's in binary form must reproduce the above copyright
941 notice,
942 //     this list of conditions and the following disclaimer in the
943 documentation
944 //     and/or other materials provided with the distribution.
945 //
946 //   * The name of Intel Corporation may not be used to endorse or promote
947 products
948 //     derived from this software without specific prior written permission.
949 //
950 // This software is provided by the copyright holders and contributors "as is"
951 and
952 // any express or implied warranties, including, but not limited to, the
953 implied
954 // warranties of merchantability and fitness for a particular purpose are
955 disclaimed.
956 // In no event shall the Intel Corporation or contributors be liable for any
957 direct,
958 // indirect, incidental, special, exemplary, or consequential damages
959 // (including, but not limited to, procurement of substitute goods or services;
960 // loss of use, data, or profits; or business interruption) however caused
961 // and on any theory of liability, whether in contract, strict liability,
962 // or tort (including negligence or otherwise) arising in any way out of
```

```
963 // the use of this software, even if advised of the possibility of such damage.
964 //
965 //  （補足条項）
966 //  このプログラムは Intel License Agreement に基づく Intel 社のオープンソ
967 ースのプログラムである。
968 //  私、飯塚岳郎が卒業研究の目的でいくつかの重要な改変を行った。
969 //  （１）ATL に基づく COM コンポーネントへの改装
970 //  （２）ファイルアクセスのデータベース化
971 //  （３）IPL 導入による BMP<->IPL（OPENCV）間のフォーマット変換機
972 能の導入
973 //
974 //  このプログラムも Intel License Agreement に準じて公開する。
975 //  なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負
976 うものとする
977 //
978 //  （本ファイルの内容）facebase.cpp
979 //  顔認識ライブラリ本体
980 //  （CFaseBase クラス, CPerson クラス, CPersonImg クラスの実装）
981 //
982 //  （本ライブラリに必要な DLL ファイル）
983 //  highgui.dll, cv.dll, cvaux.dll(OPENCV)
984 //  IPL.dll, IPLa6.dll(IPL)
985 //
986 ////////////////////////////////////////////////////////////////////////
987
988 /* プログラムに必要なインクルードファイル */
989 #include "stdafx2.h"   /* この一連の実装のインターフェース */
990 #include "facebase.h"
991 #include "direct.h"              /* 元々、サンプルコードが必要としたインター
992 フェース */
993 #include <math.h>
994 #include <float.h>
995 #include <process.h>
996
997 namespace FACEBASE
998 {
999         /* グローバル変数の実装 */
```

```
1000        CString errorText = NULL;    //エラー管理
1001
1002        /* グローバル関数の実装 */
1003        /* 既存のエラー処理機能を利用する：note1 */
1004        inline void TESTHR( HRESULT _hr )
1005        {
1006            if FAILED(_hr) _com_issue_error(_hr);
1007        }
1008
1009        /* クライアント側の原因によるデータベースエラーの表示 */
1010        HRESULT errorSQLInside( _com_error &e )
1011        {
1012            /* エラー情報の取得 */
1013            _bstr_t bstrSource(e.Source());
1014            _bstr_t bstrDescription(e.Description());
1015
1016            /* エラー情報の解析とエラー変数への出力 */
1017            errorText.AppendFormat(_T("¥nInside Error:¥n") );
1018            errorText.AppendFormat(_T("Code = %08lx¥r¥n"),
1019    e.Error() );
1020            errorText.AppendFormat(_T("Code meaning = %s¥r¥n"),
1021    e.ErrorMessage() );
1022            errorText.AppendFormat(_T("Source = %s¥r¥n"),
1023    (LPCSTR)bstrSource );
1024            errorText.AppendFormat(_T("Description = %s¥r¥n"),
1025    (LPCSTR)bstrDescription );
1026
1027            return e.Error();
1028        }
1029
1030        /* DBMS 側の原因によるデータベースエラーの表示 */
1031        void errorSQLProvider( _ConnectionPtr m_pConnection )
1032        {
1033            /* 作業用の変数 */
1034            ErrorPtr  pErr = NULL;
1035
1036            /* Connection オブジェクトより DBMS 側でのエラーを取得.
```

```
1037    */
1038                    if( (m_pConnection->Errors->Count) > 0)
1039                    {
1040                            errorText.AppendFormat(_T("Provider
1041    Error:¥r¥n") );
1042
1043                            long nCount = m_pConnection->Errors->Count;
1044                            for(long i = 0; i < nCount; i++)
1045                            {
1046                                    pErr = m_pConnection->Errors->GetItem(i);
1047                                    errorText.AppendFormat(_T("Error
1048    number: %x %s¥r¥n"),
1049                                    pErr->Number,
1050    (LPCSTR)pErr->Description );
1051                            }
1052                    }
1053            }
1054
1055            /* 配列型データに対してメモリ領域を確保
1056                以降の配列型データの処理関数にメモリ処理の一部の業務を提供
1057    */
1058            HRESULT  CopySafeArrayToHMem(  SAFEARRAY  *pvSafeArray,
1059    HANDLE *hMem )
1060            {
1061                    /* 配列型データの作成とサイズの取得 */
1062                    SAFEARRAY *psa = pvSafeArray;
1063                    ULONG ulSize = psa->rgsabound[0].cElements;
1064                    if( ulSize == 0 || pvSafeArray == NULL )
1065                    {
1066                            errorText.AppendFormat(_T("TROUBLES:与えられ
1067    た配列は空です。¥n") );
1068                            return E_POINTER;
1069                    }
1070
1071                    /* メモリ領域の確保 */
1072                    *hMem = ::GlobalAlloc( GHND, ulSize );
1073                    if( hMem == NULL ){
```

```
1074                        errorText.AppendFormat(_T("%s:%s¥n"),
1075  "AttachSafeArrayToHMem", MEMORYSHORT );
1076                        return E_OUTOFMEMORY;
1077                }
1078
1079                /* バイト型のポインタと配列を結びつける（ATTACHする）*/
1080                byte *pb;
1081                HRESULT hr = SafeArrayAccessData( psa, ( void ** )&pb );
1082                if( FAILED( hr ) )
1083                        return hr;
1084                else if( !pb )
1085                        return E_UNEXPECTED;
1086
1087                /* 確保したグローバルメモリ領域に対してデータのコピー */
1088                byte *pbStm = ( byte * )::GlobalLock( *hMem );
1089                ::memcpy( pbStm, pb, ulSize );
1090                ::GlobalUnlock( *hMem );
1091                ::SafeArrayUnaccessData( psa );
1092
1093                return S_OK;
1094        }
1095
1096        /* 配列型データに対して入出力ストリームを確立する
1097           以降の配列型データの処理関数にメモリ処理の一部の業務を提供
1098  */
1099        HRESULT        CreateStreamFromSafeArray(        SAFEARRAY
1100  *pvSafeArray, IStream **pplStream )
1101        {
1102                HRESULT hr = S_OK;
1103                HGLOBAL hMem;
1104                *pplStream = NULL;
1105
1106                // 配列型データに対してメモリ領域を確保
1107                hr = CopySafeArrayToHMem( pvSafeArray, &hMem );
1108                if( FAILED(hr))
1109                        return hr;
1110
```

```
1111                    // 確保したメモリ領域に対してストリームの作成
1112                    hr = ::CreateStreamOnHGlobal( hMem, TRUE, pplStream );
1113
1114                    return hr;
1115            }
1116
1117            /* 配列型データを OPENCV 用の画像データフォーマットに変換する
1118               データベースには配列型データとして画像データは登録されてい
1119    るため
1120               ここで JPEG->BMP->CVVIMAGE という形で画像データを変換
1121    */
1122            HRESULT SafeArrayToCvvImage( SAFEARRAY *pvSafeArray,
1123    CvvImage *newImage )
1124            {
1125                    // ここで外部の関数を呼び出す然るべき準備を執り行う
1126                    CComPtr<IStream> imageStream, imageStream2;
1127                    LPPICTURE m_pJPEG;
1128                    HRESULT hr;
1129                    HGLOBAL hMem, hMem2;
1130                    long fileSize;
1131                    HBITMAP m_hJPEGmap;
1132                    OLE_HANDLE m_pHandle;
1133
1134                    // 配列型データに対してメモリ領域を確保
1135                    hr = CopySafeArrayToHMem( pvSafeArray, &hMem );
1136                    if( FAILED(hr))
1137                            return hr;
1138                    fileSize = pvSafeArray->rgsabound[0].cElements;
1139
1140                    // CIMAGE へのデータ読み出し（JPEG 形式データを BMP 形
1141    式に変換）
1142                    CImage *imageData = new CImage();
1143                    hr = ::CreateStreamOnHGlobal( hMem, TRUE,
1144    &imageStream );
1145                    if( FAILED(hr))
1146                            return hr;
1147                    //imageData->Load( imageStream );
```

```
1148                hr = OleLoadPicture( imageStream, fileSize, FALSE,
1149    IID_IPicture, (LPVOID *)&m_pJPEG );
1150                if( FAILED(hr) || m_pJPEG == NULL )
1151                        return hr;
1152                m_pJPEG->get_Handle( &m_pHandle );
1153                imageData->Attach((HBITMAP)m_pHandle);
1154                hMem2 = ::GlobalAlloc( GHND, fileSize );
1155                hr = ::CreateStreamOnHGlobal( hMem2, TRUE,
1156    &imageStream2 );
1157                imageData->Save( imageStream2,
1158    Gdiplus::ImageFormatBMP );
1159                ::GlobalFree( hMem );
1160                //imageInfo->Save( "test041.jpg",
1161    Gdiplus::ImageFormatJPEG );
1162
1163                // ビットマップのためのポインタの定義
1164                BITMAPINFO *dib;                // pointer to bitmap
1165                RGBQUAD *rgb;                   // pointer to
1166    bitmap colors
1167                unsigned char *data;            // pointer to bitmap data
1168                BITMAPINFOHEADER *dibh;         // header
1169    beginning
1170                IplImage *img = NULL;
1171                BOOL cloneData = false;         // variable to get result
1172                int i;
1173
1174                // データ領域の展開
1175                dib = (LPBITMAPINFO)::GlobalLock( hMem2 );
1176                dibh = ( BITMAPINFOHEADER* )dib;
1177                rgb = ( RGBQUAD* )( (char*)dib +
1178    sizeof(BITMAPINFOHEADER) );
1179                data = ( unsigned char* )( (char*)rgb + sizeof(RGBQUAD) *
1180    256 );
1181
1182                // 画像ヘッダ部分の定義
1183                dibh->biSize = sizeof(BITMAPINFOHEADER);
1184                dibh->biWidth = imageData->GetWidth();
```

```
1185                    dibh->biHeight = imageData->GetHeight();
1186                    dibh->biPlanes = 1;
1187                    dibh->biBitCount = 8;
1188                    dibh->biCompression = BI_RGB;
1189                    dibh->biSizeImage                                       =
1190    pvSafeArray->rgsabound[0].cElements;
1191                    dibh->biClrUsed = 256;
1192                    dibh->biClrImportant = 0;
1193
1194                    // カラーパレット部分のコピー
1195                    for( i = 0; i < 256; i++ )
1196                            rgb[i].rgbBlue = rgb[i].rgbGreen = rgb[i].rgbRed =
1197    (unsigned char)i;
1198
1199                    // IPL イメージの作成
1200                    img = iplTranslateDIB( dibh, &cloneData );
1201                    IplROI    roi;
1202                    img->roi = &roi;
1203                    roi = RectToROI( CRect( 15, 0, imageData->GetWidth() + 15,
1204    imageData->GetHeight() ) );
1205
1206                    // CVVIMAGE インスタンスの作成
1207                    newImage->CopyOf( img );
1208
1209                    ::GlobalUnlock( hMem2 );
1210                    ::GlobalFree( hMem2 );
1211                    return S_OK;
1212            }
1213
1214            /* OPENCV 用の画像データを配列型データに変換する
1215                データベースに配列型データとして画像データを保存するために
1216    利用
1217                ここで CVVIMAGE->BMP->JPEG という形で画像データを変換
1218    */
1219            SAFEARRAY *SafeArrayFromCvvImage( CvvImage *newImage )
1220            {
1221                    CComPtr<IStream> imageStream;
```

```
1222                HRESULT hr;
1223                int i;
1224
1225                //コピーするべきデータの確保
1226                IplImage *image = newImage->GetImage();
1227                if( image->imageSize <= 0 )
1228                        return NULL;
1229
1230                //データを格納する BYTE 型のメモリ領域の確保
1231                int dataSize = image->imageSize + sizeof(RGBQUAD) * 256
1232  + sizeof(BITMAPINFOHEADER) + 14;
1233                HGLOBAL hMem = ::GlobalAlloc( GHND, dataSize );
1234                if( hMem == NULL )
1235                {
1236                        errorText.AppendFormat(_T("%s:%s¥n"),
1237  "AttachSafeArrayToHMem", MEMORYSHORT );
1238                        return NULL;
1239                }
1240                BITMAPINFOHEADER        *bMapHead       =
1241  ( BITMAPINFOHEADER * )::GlobalLock( hMem );
1242
1243                // コピー先データ領域の展開
1244                BITMAPINFOHEADER        *dibh           =
1245  ( BITMAPINFOHEADER* )bMapHead;
1246                RGBQUAD *rgb = ( RGBQUAD* )( (char*)bMapHead +
1247  sizeof(BITMAPINFOHEADER) );
1248
1249                // コピー先画像ヘッダ部分の定義
1250                dibh->biSize = sizeof(BITMAPINFOHEADER);
1251                dibh->biWidth = image->width;
1252                dibh->biHeight = image->height;
1253                dibh->biPlanes = 1;
1254                dibh->biBitCount = 8;
1255                dibh->biCompression = BI_RGB;
1256                dibh->biSizeImage = dataSize;
1257                dibh->biClrUsed = 256;
1258                dibh->biClrImportant = 0;
```

```
1259
1260                    // カラーパレット部分のコピー
1261                    for( i = 0; i < 256; i++ )
1262                            rgb[i].rgbBlue = rgb[i].rgbGreen = rgb[i].rgbRed =
1263    (unsigned char)i;
1264
1265                    // 画像データのコピー(白黒画像仕様)
1266                    HDC hdc = (HDC)GetDC(NULL);
1267                    LPBYTE pBits;
1268                    HBITMAP m_hBitmap = CreateDIBSection( hdc,
1269    (BITMAPINFO*)bMapHead, DIB_RGB_COLORS, (VOID **)&pBits, NULL,
1270    0 );
1271                    iplConvertToDIBSep( image, bMapHead, (char *)pBits,
1272    IPL_DITHER_NONE, IPL_PALCONV_NONE );
1273
1274                    //データを格納する BYTE 型のテンポラリの宣言
1275                    HGLOBAL hMem1 = ::GlobalAlloc( GHND, dataSize );
1276                    hr = ::CreateStreamOnHGlobal( hMem1, TRUE,
1277    &imageStream );
1278
1279                    //変換した結果のメモリ上への書き込み
1280                    CImage *imageInfo = new CImage();
1281                    imageInfo->Attach(m_hBitmap);
1282                    if( FAILED(hr) )
1283                            errorText.AppendFormat(_T("ERROR IN
1284    imageStream¥n") );
1285                    hr = imageInfo->Save( imageStream,
1286    Gdiplus::ImageFormatJPEG );
1287                    imageInfo->Detach();
1288                    DeleteObject( m_hBitmap );
1289                    ::GlobalFree( hMem );
1290
1291                    //データを格納する BYTE 型の SafeArray 配列の宣言
1292    77878(TARGET) 77864(NOW)
1293                    SAFEARRAY FAR *m_pImageData;
1294                    SAFEARRAYBOUND rgsabound[1];
1295                    rgsabound[0].lLbound = 0;
```

```
1296                    rgsabound[0].cElements = dataSize + 1;
1297                    m_pImageData = SafeArrayCreate(VT_UI1,1,rgsabound);
1298                    if( m_pImageData == NULL )
1299                            return NULL;
1300
1301                    // バイト型のポインタと配列を結びつける（ATTACH する）
1302                    byte *pb;
1303                    hr = SafeArrayAccessData( m_pImageData, ( void ** )&pb );
1304                    if( FAILED( hr ) )
1305                            return NULL;
1306                    else if( !pb )
1307                            return NULL;
1308
1309                    //変換した結果の配列への書き込み
1310                    byte *pb2 = ( byte * )::GlobalLock( hMem1 );
1311                    ::memcpy( pb, pb2, dataSize );
1312                    ::GlobalUnlock( hMem1 );
1313                    ::GlobalFree( hMem1 );
1314                    ::SafeArrayUnaccessData( m_pImageData );
1315
1316                    CvvImage *testImage = new CvvImage();
1317                    hr = SafeArrayToCvvImage( m_pImageData, testImage );
1318                    if( FAILED(hr))
1319                            errorText.AppendFormat(_T("ERROR                IN
1320    Final¥n") );
1321
1322                    return m_pImageData;
1323            }
1324
1325
1326            /* 輝度の正規化（8 ビット白黒画像に正規化） */
1327            void NormalizeIntensity(IplImage* ipl_image, int level)
1328            {
1329                    CvSize roi;
1330                    int step;
1331                    uchar* img;
1332
```

```
1333                ASSERT( (level>0) && (level <=255) );
1334                cvGetImageRawData(ipl_image, &img, &step, &roi);
1335
1336                int width = roi.width;
1337                int height = roi.height;
1338
1339                int mean = (int)cvMean( ipl_image );
1340                // normalize to 128
1341                for( int i = 0; i < height; i++, img+=step )
1342                {
1343                        for( int j = 0; j < width; j++ )
1344                        {
1345                                int newval = img[j] + level - mean;
1346                                newval = (newval < 0) ? 0 : newval;
1347                                newval = (newval > 255) ? 255 : newval;
1348                                img[j] = newval;
1349                        }
1350                }
1351        }
1352
1353        /* 顔認識エンジン内部で利用される関数 */
1354        /* HMM 処理のための画像の正規化 */
1355        void NormalizeImageForHMM( IplImage* ipl_scaled, IplImage*
1356 normalized_image )
1357        {
1358                //iplFixedFilter( ipl_scaled, normalized_image,
1359 IPL_SOBEL_3x3_H );
1360        }
1361
1362        void ExtractDCT( float* src, float* dst, int num_vec, int dst_len )
1363        {
1364                float* src_ = src+1;
1365                float* dst_ = dst;
1366
1367                for( int i = 0; i < num_vec; i++ )
1368                {
1369                        memcpy( dst_, src_, dst_len * sizeof(float) );
```

```
1370                        src_+= dst_len+1;
1371                        dst_+= dst_len;
1372                }
1373        }
1374
1375        /*****************************************************************
1376 ***************************¥
1377        *                                          CBaseInfo   class
1378        *   このクラスは本ライブラリで利用するデータベースに関する情報
1379 を一括管理する
1380        *   本ライブラリで必要なデータベース関係のうちの一部を担当する
1381        ¥*****************************************************************
1382 ****************************/
1383        /* デストラクタ　このオブジェクトのガベージコレクションが行われ
1384 た時、
1385           自動的にデータベース接続を完了する*/
1386        CBaseInfo::~CBaseInfo()
1387        {
1388                m_pConnection->Close();
1389        }
1390
1391        /* 接続先情報の設定 */
1392        HRESULT   CBaseInfo::SetDBInfo( const CString& PROVIDER,
1393                                        const CString& SOURCE,
1394                                        const CString& DB,
1395                                        const CString& USER,
1396                                        const CString& PASSWD )
1397        {
1398                /* 異常な入力に対する処理 */
1399                if( PROVIDER.GetLength() == 0 || SOURCE.GetLength()
1400 == 0
1401                        || DB.GetLength() == 0 || USER.GetLength() ==
1402 0 )
1403                        return CTL_E_INVALIDPROPERTYVALUE;
1404
1405                /* 接続先情報の設定 */
1406                m_Provider = PROVIDER;
```

```
1407                    m_DataSource = SOURCE;
1408                    m_dbName = DB;
1409                    m_userName = USER;
1410                    m_passWord = PASSWD;
1411
1412                    return S_OK;
1413            }
1414
1415            /* テーブル名の設定 */
1416            HRESULT   CBaseInfo::SetTABLEInfo( const CString& info, const
1417    CString& image, const CString& config )
1418            {
1419                    /* 異常な入力に対する処理 */
1420                    if( info.GetLength() == 0 || image.GetLength() == 0
1421                                    || config.GetLength() == 0 )
1422                            return CTL_E_INVALIDPROPERTYVALUE;
1423
1424                    /* 値の設定 */
1425                    m_configBase = config;
1426                    m_infoBase = info;
1427                    m_imageBase = image;
1428
1429                    return S_OK;
1430            }
1431
1432            // データベース接続を作成するのに必要な接続文字列を作成
1433            CString CBaseInfo::makeup_destination()
1434            {
1435                    CString destination;
1436
1437                    destination.AppendFormat(_T("Provider=%s;"),
1438    (LPCSTR)m_Provider );
1439                    destination.AppendFormat(_T("Data              Source=%s;"),
1440    (LPCSTR)m_DataSource );
1441                    destination.AppendFormat(_T("Initial          Catalog=%s;"),
1442    (LPCSTR)m_dbName );
1443                    destination.AppendFormat(_T("User                 ID=%s;"),
```

```
(LPCSTR)m_userName );
                destination.AppendFormat(_T("pwd=%s;"),
(LPCSTR)m_passWord );
                return destination;
        }

        /* データベース接続を開始する */
        HRESULT CBaseInfo::openDB()
        {
                /* 作業用インスタンスの設定 */
                m_pConnection = _ConnectionPtr("ADODB.Connection");
                _bstr_t strMissing(L"");
                HRESULT hr;

                /* DB への接続を試みる */
                /* SQL Server 名 + ユーザー名 + パスワード + 使用するデ
ータベース名 */
                CString destination = makeup_destination();
                try
                {

        TESTHR(m_pConnection.CreateInstance("ADODB.Connection"));
                        m_pConnection->Open(        (LPCSTR)destination,
strMissing,"",
                                adConnectUnspecified );
                        return S_OK;
                }
                catch( _com_error &e )
                {
                        hr = errorSQLInside( e );
                        errorSQLProvider( m_pConnection );
                        return hr;
                }
        }

        /*****************************************************************
**************************¥
```

```
1481        *                                               CFaceBase   class
1482        *   このクラスは本ライブラリの顔認識機能を一括管理するインター
1483    フェースとして機能する
1484        *   顔認識機能を実装したクラスの呼び出し、データベースのアクセス
1485    などを担当する
1486        ¥**************************************************************
1487    ***************************/
1488        ////////////////////////////////////////////////////////////////
1489        // コンストラクタとデストラクタ関係
1490        ////////////////////////////////////////////////////////////////
1491        /* コンストラクタ */
1492        CFaceBase::CFaceBase()
1493        {
1494                //default parameters
1495                m_stnum[0] = 5;
1496                m_stnum[1] = 3;
1497                m_stnum[2] = 6;
1498                m_stnum[3] = 6;
1499                m_stnum[4] = 6;
1500                m_stnum[5] = 3;
1501                for( int i = 0; i < 128; i++ )
1502                {
1503                        m_mixnum[i] = 3;
1504                }
1505
1506                m_modified = false;
1507                m_trained_index = -1;
1508                SetImageSize( CSize(100,120) );
1509
1510                m_delta = cvSize(4,4);
1511                m_obsSize = cvSize(3,3);
1512                m_dctSize = cvSize(12,12);
1513
1514                m_useWidth = FALSE;
1515                m_useHeight = FALSE;
1516                m_scaleWidth = 0;
1517                m_scaleHeight = 0;
```

```
1518                m_suppress_intensity = FALSE;
1519
1520                image_load = false;
1521        }
1522
1523        /* デストラクタ */
1524        CFaceBase::~CFaceBase()
1525        {
1526                Unload();
1527        }
1528
1529        /* 顔認識情報を保存したデータベース接続先の設定 */
1530        HRESULT   CFaceBase::SetDBInfo( const CString& PROVIDER,
1531                                        const CString& SOURCE,
1532                                        const CString& DB,
1533                                        const CString& USER,
1534                                        const CString& PASSWD )
1535        {
1536                HRESULT hr = baseInfo.SetDBInfo( PROVIDER, SOURCE,
1537  DB, USER, PASSWD );
1538                if( FAILED(hr) )
1539                        return hr;
1540                SetModified();
1541                return hr;
1542        }
1543
1544        /* 顔認識情報を保存したテーブル名の設定 */
1545        HRESULT   CFaceBase::SetTABLEInfo( const CString& info, const
1546  CString& image, const CString& config )
1547        {
1548                HRESULT   hr   =   baseInfo.SetTABLEInfo(   info,   image,
1549  config );
1550
1551                if( FAILED(hr) )
1552                        return hr;
1553                SetModified();
1554                return hr;
```

```
1555        }
1556
1557        ////////////////////////////////////////////////////////////////////
1558        // 顔認識エンジンの設定情報関係
1559        ////////////////////////////////////////////////////////////////////
1560        /* 顔認識エンジンの設定情報を読み込む　*/
1561        /* 顔認識エンジンを利用するには絶対不可欠な処理であり、最初に行
1562  う処理である　*/
1563        HRESULT CFaceBase::load_config()
1564        {
1565                //CoInitialize(NULL);
1566
1567                /* 作業用の変数　*/
1568                CvSize dctSize, delta, obsSize;
1569                int value;
1570                int numbers[8];
1571                int NumMix;
1572                int FixedWidth;
1573                bool m_useWidth;
1574                int FixedHeight;
1575                bool m_useHeight;
1576                bool SuppressIntensity;
1577
1578                /* 作業用インスタンスの設定　*/
1579                HRESULT hr = baseInfo.openDB();
1580                if( FAILED( hr ) )
1581                        return hr;
1582                _ConnectionPtr                m_pConnection                =
1583  baseInfo.get_ConnectionPtr();
1584                CString temp;
1585
1586                /*        データベースへのアクセス（失敗する可能性があるの
1587  で TRY 文で囲む）　*/
1588                try
1589                {
1590                        /* データベースと接続　*/
1591                        CString basename = baseInfo.ConfigBaseName();
```

```
1592                        _RecordsetPtr m_pRecordSet("ADODB.Recordset");
1593
1594        TESTHR(m_pRecordSet.CreateInstance(__uuidof(Recordset)));
1595                        m_pRecordSet->Open(          (LPCSTR)basename,
1596  _variant_t((IDispatch *)m_pConnection, true),
1597                                                   adOpenStatic,
1598                                                   adLockReadOnly,
1599                                                   adCmdTable);
1600
1601                        /* 基本的なパラメータの設定  */
1602                        m_useWidth = FALSE;
1603                        FixedWidth = 0;
1604                        m_useHeight = FALSE;
1605                        FixedHeight = 0;
1606                        SuppressIntensity = FALSE;
1607
1608                        /*  画像関係の設定の読込  */
1609                        m_pRecordSet->MoveFirst();
1610                        dctSize.height                                    =
1611  m_pRecordSet->Fields->Item["WINDOW_HEIGHT"]->GetValue();
1612                        dctSize.width                                     =
1613  m_pRecordSet->Fields->Item["WINDOW_WIDTH"]->GetValue();
1614                        delta.width                                       =
1615  m_pRecordSet->Fields->Item["DELTA_X"]->GetValue();
1616                        delta.height                                      =
1617  m_pRecordSet->Fields->Item["DELTA_Y"]->GetValue();
1618                        obsSize.width                                     =
1619  m_pRecordSet->Fields->Item["DCT_COEFF_X"]->GetValue();
1620                        obsSize.height                                    =
1621  m_pRecordSet->Fields->Item["DCT_COEFF_Y"]->GetValue();
1622
1623                        /* HMM（隠れマルコフモデル）関係の読込  */
1624                        NumMix                                            =
1625  m_pRecordSet->Fields->Item["NUM_MIXTURE"]->GetValue();
1626                        numbers[0]                                        =
1627  m_pRecordSet->Fields->Item["SUPER_STATES"]->GetValue();
1628
```

```
1629                        /* HMM（隠れマルコフモデル）関係の読込：状態ベク
1630    トルの出力　*/
1631                        CString                 tempString                  =
1632    m_pRecordSet->Fields->Item["STATE"]->GetValue();
1633                        CString resToken;
1634                        int curPos = 0;
1635                        resToken = tempString.Tokenize(" ",curPos);
1636                        for( int i = 1; i <= numbers[0] || resToken != ""; i++ )
1637                        {
1638                                sscanf( resToken, "%d ", &numbers[i] );
1639                                resToken    =    tempString.Tokenize(    "    ",
1640    curPos );
1641                        }
1642
1643                        /* その他の設定の読込　*/
1644                        FixedWidth                                              =
1645    (int)m_pRecordSet->Fields->Item["FIXED_WIDTH"]->GetValue();
1646                        if( FixedWidth != 0 )
1647                                m_useWidth = TRUE;
1648                        FixedHeight                                             =
1649    (int)m_pRecordSet->Fields->Item["FIXED_HEIGHT"]->GetValue();
1650                        if( FixedHeight != 0 )
1651                                m_useHeight = TRUE;
1652
1653                        /* 獲得したデータの正当性のチェック　*/
1654                        if    (    (dctSize.width    <    obsSize.width)    ||
1655    (dctSize.height < obsSize.height) )
1656                        {
1657                                errorText.AppendFormat(_T("Wrong
1658    parameters in config file.¥n") );
1659                                errorText.AppendFormat(_T("New
1660    parameters were not loaded.¥n") );
1661                                return E_INVALIDARG;
1662                        }
1663
1664                        /* 顔認識エンジン FACEBASE（つまりこのクラスの
1665    インスタンスに）にデータを設定　*/
```

```
1666                    /* 設定用の専用の関数を利用して行う　*/
1667                    i = SetParams(//sampling params
1668                                                    dctSize,
1669                                                    obsSize,
1670                                                    delta,
1671                                                    //HMM params
1672                                                    numbers,
1673                                                    NumMix,
1674                                                    //image          scaling
1675  params
1676                                                    m_useWidth,
1677                                                    FixedWidth,
1678                                                    m_useHeight,
1679                                                    FixedHeight,
1680                                                    //intensity suppression
1681                                                    SuppressIntensity,
1682  TRUE );
1683                    if( i <= 0 )
1684                            return CTL_E_INVALIDPROPERTYVALUE;
1685            }
1686            catch( _com_error &e )          /* エラー処理 */
1687            {
1688                    errorSQLProvider( m_pConnection );
1689                    return errorSQLInside( e );
1690            }
1691
1692            //::CoUninitialize();
1693            return S_OK;
1694    }
1695
1696    /* 顔認識エンジンの設定情報を保存する　*/
1697    HRESULT CFaceBase::save_config()
1698    {
1699            //CoInitialize(NULL);
1700
1701            /* 作業用インスタンスの設定　*/
1702            HRESULT hr = baseInfo.openDB();
```

```
1703                    if( FAILED( hr ) )
1704                            return hr;
1705                    _ConnectionPtr                  m_pConnection                   =
1706    baseInfo.get_ConnectionPtr();
1707                    CString temp;
1708
1709                    /*      データベースへのアクセス（失敗する可能性があるの
1710    で TRY 文で囲む）    */
1711                    try
1712                    {
1713                            /* データベースと接続   */
1714                            CString basename = baseInfo.ConfigBaseName();
1715                            _RecordsetPtr m_pRecordSet("ADODB.Recordset");
1716
1717            TESTHR(m_pRecordSet.CreateInstance(__uuidof(Recordset)));
1718                            m_pRecordSet->Open(             (LPCSTR)basename,
1719    _variant_t((IDispatch *)m_pConnection, true),
1720                                                            adOpenStatic,
1721                                                            adLockOptimistic,
1722                                                            adCmdTable);
1723
1724                            /* 画像関係の設定を出力   */
1725
1726            m_pRecordSet->Fields->Item["WINDOW_HEIGHT"]->PutValue( m
1727    _dctSize.height );
1728
1729            m_pRecordSet->Fields->Item["WINDOW_WIDTH"]->PutValue( m_
1730    dctSize.width );
1731
1732            m_pRecordSet->Fields->Item["DELTA_X"]->PutValue( m_delta.wid
1733    th );
1734
1735            m_pRecordSet->Fields->Item["DELTA_Y"]->PutValue( m_delta.heig
1736    ht );
1737
1738            m_pRecordSet->Fields->Item["DCT_COEFF_X"]->PutValue( m_obs
1739    Size.width );
```

```
1740
1741            m_pRecordSet->Fields->Item["DCT_COEFF_Y"]->PutValue(  m_obs
1742    Size.height );
1743
1744                            /* HMM（隠れマルコフモデル）関係の設定を出力   */
1745
1746            m_pRecordSet->Fields->Item["NUM_MIXTURE"]->PutValue(  m_m
1747    ixnum[0] );
1748
1749            m_pRecordSet->Fields->Item["SUPER_STATES"]->PutValue(  m_st
1750    num[0] );
1751                            for(int i = 1; i <= m_stnum[0]; i++ )
1752                            {
1753                                    temp.AppendFormat(_T("%d "), m_stnum[i] );
1754                            }
1755
1756            m_pRecordSet->Fields->Item["STATE"]->PutValue(  _variant_t((LP
1757    CSTR)temp) );
1758
1759                            /*  その他の設定を出力   */
1760                            if( m_useWidth == TRUE )
1761
1762            m_pRecordSet->Fields->Item["FIXED_WIDTH"]->PutValue( m_scal
1763    eWidth );
1764                            if( m_useHeight == TRUE )
1765
1766            m_pRecordSet->Fields->Item["FIXED_HEIGHT"]->PutValue(  m_sc
1767    aleHeight );
1768                            if( m_suppress_intensity )
1769
1770            m_pRecordSet->Fields->Item["SUPPRESS_INTENSITY"]->PutVal
1771    ue( 1 );
1772
1773                            /*  データベースの更新を実施   */
1774                            m_pRecordSet->Update();
1775                    }
1776                    catch( _com_error &e )          /*  エラー処理  */
```

```
1777                {
1778                        errorSQLProvider( m_pConnection );
1779                        return errorSQLInside( e );
1780                }
1781
1782                //::CoUninitialize();
1783                return S_OK;
1784        }
1785
1786        ////////////////////////////////////////////////////////////////////
1787        // 顔認識テーブル、画像テーブルへのアクセス
1788        ////////////////////////////////////////////////////////////////////
1789        /* 顔認識情報を格納したデータベースの読み込み
1790           ユースケース記述では「個人の認識」部分で、顔認識を行う前に読
1791   込
1792           個人情報クラス（CPPERSON）に必要なデータをこちらで一括処
1793   理・登録　*/
1794        HRESULT　CFaceBase::Load()
1795        {
1796                //CoInitialize(NULL);
1797
1798                /* データベース接続の設定　*/
1799                HRESULT hr;
1800                _ConnectionPtr                m_pConnection                =
1801   baseInfo.get_ConnectionPtr();
1802
1803                /* CPersonList（データ保管用の個人データのリスト構造）の
1804   クリア　*/
1805                m_base.RemoveAll();
1806
1807                /*        データベースへのアクセス（失敗する可能性があるの
1808   で TRY 文で囲む）　*/
1809                try
1810                {
1811                        /* パターン情報とその他の情報を格納したテーブルと
1812   接続　*/
1813                        CString basename = baseInfo.BaseName();
```

```
1814                        _RecordsetPtr m_pRecordSet("ADODB.Recordset");
1815
1816        TESTHR(m_pRecordSet.CreateInstance(__uuidof(Recordset)));
1817                        hr  =  m_pRecordSet->Open(  (LPCSTR)basename,
1818  _variant_t((IDispatch *)m_pConnection, true),
1819                                                    adOpenStatic,
1820                                                    adLockReadOnly,
1821                                                    adCmdTable);
1822                        if( FAILED(hr))
1823                                return hr;
1824
1825                        /* 顔写真を格納したテーブルと接続  */
1826                        basename = baseInfo.ImageBaseName();
1827                        _RecordsetPtr
1828  m_pRecordSet_image("ADODB.Recordset");
1829                        TESTHR(                     m_pRecordSet_image.
1830  CreateInstance(__uuidof(Recordset)));
1831                        hr                                              =
1832  m_pRecordSet_image->Open(  (LPCSTR)basename,  _variant_t((IDispatch
1833  *)m_pConnection, true),
1834                                                    adOpenStatic,
1835                                                    adLockReadOnly,
1836                                                    adCmdTable);
1837                        if( FAILED(hr))
1838                                return hr;
1839                        errorText.AppendFormat(_T("データベース接続成
1840  功:¥n") );
1841
1842                        /* データの読込  */
1843                        m_pRecordSet->MoveFirst();
1844                        while( !m_pRecordSet->EndOfFile )
1845                        {
1846                                /* 個人情報テーブルと接続するのに必要な情
1847  報の読み込み  */
1848                                CPerson *person = new CPerson( this );
1849                                person->m_PersonID                      =
1850  m_pRecordSet->GetCollect(L"PersonID");
```

```
1851                                    person->m_name                                  =
1852    m_pRecordSet->GetCollect(L"Name");
1853
1854                                    /* HMM モデルによる顔認識に必要な認識パタ
1855    ーン情報の読み込み　*/
1856                                    CString                 temp                    =
1857    m_pRecordSet->GetCollect(L"HMM");
1858                                    if(    temp.CompareNoCase("NULL")    !=    0
1859    && !temp.IsEmpty() )
1860                                    {
1861                                            person->m_hmm           =           new
1862    CContEHMM();
1863
1864            if( person->m_hmm->Load((LPCSTR)temp) )
1865                                                    person->m_trained = true;
1866
1867            errorText.Append( person->m_hmm->errorText );
1868                                    }
1869
1870                                    /* 顔写真情報の読み込み　*/
1871                                    if( image_load == true )
1872
1873            person->Load( m_pRecordSet_image );
1874
1875                                    /* データ保管用の個人データのリスト構造に
1876    個人データの付加 */
1877                                    person->SetModified( false );           //      重
1878    要：保存側での効率に大きく影響
1879                                    person->m_trained = true;
1880                                    if( person->m_PersonID < 0 )
1881                                    {
1882                                            delete person;
1883
1884            errorText.AppendFormat(_T("FAILED    TO    ADD    PERSON    TO
1885    CPERSONLIST.¥n") );;
1886                                            continue;
1887                                    }
```

```
1888                                m_base.AddTail( person );
1889                                m_pRecordSet->MoveNext();
1890                        }
1891                }
1892                catch( _com_error &e )          /* エラー処理 */
1893                {
1894                        errorSQLProvider( m_pConnection );
1895                        return errorSQLInside( e );
1896                }
1897
1898                /* 終了処理 */
1899                //::CoUninitialize();
1900                return S_OK;
1901        }
1902
1903        /* 顔認識エンジン（つまりクラス）の終了処理を実装
1904           顔認識の学習成果と顔認識エンジンの設定情報を保存
1905           ユースケース記述の「個人情報の検索」部分の最後に用いる */
1906        void   CFaceBase::Unload()
1907        {
1908                Save();                         /* 顔認識情報を保存 */
1909                save_config();          /* 顔認識エンジンの設定情報を保存
1910   */
1911
1912                while( !m_base.IsEmpty() )
1913                {
1914                        CPerson* person = m_base.RemoveHead();
1915                        delete person;
1916                }
1917        }
1918
1919        /* 顔認識情報を格納したデータベースへの現在の情報の保存   */
1920        /* 顔認識、パターン学習の成果をこのメソッドを利用してデータベー
1921   スに保存   */
1922        HRESULT   CFaceBase::Save()
1923        {
1924                _ConnectionPtr m_pConnection;
```

```
1925                HRESULT hr;
1926                bool is_eof = false;
1927
1928                //CoInitialize(NULL);
1929                /* 何らかの形で顔認識、パターン学習が行われた場合のみこの
1930 メソッドは機能  */
1931                if( IsModified() )
1932                {
1933                        /*      データベースへのアクセス（失敗する可能性が
1934 あるので TRY 文で囲む）   */
1935                        try
1936                        {
1937
1938                                /* データベース接続の設定  */
1939                                hr = baseInfo.openDB();
1940                                if( FAILED( hr ) )
1941                                        return hr;
1942                                _ConnectionPtr        m_pConnection        =
1943 baseInfo.get_ConnectionPtr();
1944
1945                                /* パターン情報とその他の情報を格納したテ
1946 ーブルと接続  */
1947                                CString basename = baseInfo.BaseName();
1948                                _RecordsetPtr
1949 m_pRecordSet("ADODB.Recordset");
1950
1951          TESTHR(m_pRecordSet.CreateInstance(__uuidof(Recordset)));
1952                                m_pRecordSet->Open(   (LPCSTR)basename,
1953 _variant_t((IDispatch *)m_pConnection, true),
1954                                                        adOpenStatic,
1955                                                        adLockOptimistic,
1956                                                        adCmdTable);
1957                                if( FAILED(hr))
1958                                        return hr;
1959
1960                                /* 顔写真を格納したテーブルと接続  */
1961                                basename = baseInfo.ImageBaseName();
```

```
                                _RecordsetPtr
m_pRecordSet_image("ADODB.Recordset");

        TESTHR(m_pRecordSet_image.CreateInstance(__uuidof(Recordset)
));
                                hr                                      =
m_pRecordSet_image->Open(  (LPCSTR)basename,  _variant_t((IDispatch
*)m_pConnection, true),
                                                        adOpenStatic,

        adLockOptimistic,
                                                        adCmdTable);
                                if( FAILED(hr))
                                        return hr;

                                /* データ保管用の個人データのリスト構造の
内容を読み出しては保存 */
                                POSITION pos = m_base.GetHeadPosition();
                                m_pRecordSet->MoveFirst();
                                while( true )
                                {
                                        /* 書き込み位置へ移動、変更不要なも
のはパス   */
                                        CPerson*            person          =
m_base.GetNext( pos );
                                        if( !pos )
                                                break;

                                        /* 新規レコードの作成(追加があった
場合)   */
                                        if( m_pRecordSet->EndOfFile )
                                        {
                                                /* データベースへのレコード
挿入   */
                                                hr                      =
m_pRecordSet->AddNew();
                                                if( FAILED(hr))
```

```
1999                                                    {
2000
2001            errorText.AppendFormat(_T("新規レコード付加失敗¥n"));
2002                                                            return hr;
2003                                                    }
2004                                                    errorText.AppendFormat(_T("
2005    データベース更新成功¥n"));
2006                                            }
2007
2008                                            /* 認識パターンの修正結果の登録   */
2009
2010            //m_pRecordSet->Fields->Item["PERSONID"]->PutValue(   person->
2011    GetPersonID() );
2012
2013            m_pRecordSet->Fields->Item["NAME"]->PutValue(   _variant_t((LP
2014    CSTR)person->GetName()) );
2015                                            if( person->m_hmm == NULL )
2016                                            {
2017
2018            m_pRecordSet->Fields->Item["HMM"]->PutValue( "NULL" );
2019                                            }
2020                                            else
2021                                            {
2022                                                    CString    *temp    =    new
2023    CString();
2024
2025            if( !person->m_hmm->Save( temp ) )
2026
2027            m_pRecordSet->Fields->Item["HMM"]->PutValue( "NULL" );
2028                                                    else
2029
2030            m_pRecordSet->Fields->Item["HMM"]->PutValue(   (LPCSTR)(*tem
2031    p) );
2032
2033            errorText.Append( person->m_hmm->errorText );
2034                                            }
2035
```

```
2036                                    /* 新たに得られた顔写真の登録　*/
2037
2038            person->Save( m_pRecordSet_image );
2039                                    person->SetModified( false );
2040
2041                                    /* 顔認識テーブル上の書き込み位置移
2042    動　*/
2043                                    if( !m_pRecordSet->EndOfFile )
2044                                            m_pRecordSet->MoveNext();
2045                                }
2046
2047                                /* データベースの更新を実施　*/
2048                                hr = m_pRecordSet->Update();
2049                                if( FAILED(hr))
2050                                {
2051                                        errorText.AppendFormat(_T("新規レ
2052    コード付加失敗¥n"));
2053                                        _com_error e = _com_error(hr);
2054                                        errorText.AppendFormat(_T("エラー
2055    原因：%s¥n"), e.ErrorMessage() );
2056                                        return hr;
2057                                }
2058                                errorText.AppendFormat(_T("データベース更
2059    新完了¥n"));
2060                        }
2061                        catch( _com_error &e )          /* エラー処理 */
2062                        {
2063                                m_pConnection->RollbackTrans();
2064                                errorSQLProvider( m_pConnection );
2065                                return errorSQLInside( e );
2066                        }
2067
2068                        /* データベースが更新済であることを設定　*/
2069                        SetModified(false);
2070                        //return S_OK;
2071                }
2072                //::CoUninitialize();
```

```
2073                    return S_OK;
2074            }
2075
2076            /* データ構造から登録された個人を削除
2077                このメソッドは顔情報データベースからある個人を削除する場合
2078    に用いる
2079                今回のアプリケーションでは利用されないが、完全な実装には必要
2080    */
2081            void    CFaceBase::RemovePerson( const long& PersonID )
2082            {
2083                    POSITION pos = m_base.GetHeadPosition();
2084                    CPerson *person;
2085
2086                    while( pos )
2087                    {
2088                            person = m_base.GetNext(pos);
2089                            if(     person      &&      (       PersonID      ==
2090    person->GetPersonID() ) ) break;
2091                    }
2092
2093                    if( person )
2094                    {
2095                            person->Unload();
2096                            m_base.RemoveAt( pos );
2097                            delete person;
2098                            SetModified();
2099                    }
2100            }
2101
2102    //////////////////////////////////////////////////////////////////
2103    // CPERSONLIST 型（データ保管用に設定された個人データのリスト構造）
2104    へのアクセスの実装
2105    // リスト構造というのはデータ構造の一種で、オブジェクト指向の言語では
2106    // データ処理のプログラムごとライブラリとして提供される。
2107    //////////////////////////////////////////////////////////////////
2108            /* インデックスより個人情報データ（CPERSON 型）を獲得
2109                もともと提供されたプログラムがリスト構造に登録された個人情
```

```
報の検索用に利用 */
        CPerson*    CFaceBase::GetPerson( const int& index )
        {
                ASSERT( index >= 0 );
                POSITION pos = m_base.FindIndex( index );
                return pos ? m_base.GetAt( pos ) : 0;
        }

        /* 顔認識テーブルの主キーより個人情報データ（CPERSON 型）を獲
得
            リスト構造に登録された個人情報の検索のため最も多用される */
        CPerson*        CFaceBase::FindPersonByPersonID(    const    long&
PersonID )
        {
                POSITION pos = m_base.GetHeadPosition();

                while( pos )
                {
                        CPerson* person = m_base.GetNext(pos);
                        if(        person        &&        (        PersonID        ==
person->GetPersonID() ) ) return person;
                }
                return 0;
        }

        /* 顔認識テーブルの名前データより個人情報データ（CPERSON 型）
を獲得
            顔認識テーブルの名前データはもともとあったデータだが、現在は
利用していない
            従って、プログラム内からは利用されない。 */
        CPerson*        CFaceBase::FindPersonByName(    const    CString&
name )
        {
                POSITION pos = m_base.GetHeadPosition();

                while( pos )
                {
```

```
2147                        CPerson* person = m_base.GetNext(pos);
2148                        if(                person                &&
2149  name.CompareNoCase( person->GetName()) == 0 ) return person;
2150                }
2151                return 0;
2152        }
2153
2154        /* 個人情報データよりデータ保管用に設定された個人データのリスト
2155  構造のインデックスを獲得
2156            この関数は FindPersonByPersonID 関数、FindPersonByName 関
2157  数で利用される */
2158        int   CFaceBase::GetPersonIndex( CPerson* person )
2159        {
2160                POSITION pos = m_base.GetHeadPosition();
2161                int i;
2162
2163                for( i = 0; pos != 0; i++ )
2164                {
2165                        CPerson* p = m_base.GetNext(pos);
2166                        if( p == person ) return i;
2167                }
2168
2169                return -1;
2170        }
2171
2172        /* データ構造内部の HMMInfo（隠れマルコフ型の学習パターン情報）
2173  のクリア
2174            パターン学習の全面やり直しが必要になった場合に利用する */
2175        void   CFaceBase::DeleteHMMInfo()
2176        {
2177                POSITION pos = m_base.GetHeadPosition();
2178
2179                while( pos )
2180                {
2181                        CPerson* person = m_base.GetNext(pos);
2182                        if( person )
2183                        {
```

```
2184                                    person->DeleteHMMInfo();
2185                            }
2186                    }
2187            }
2188
2189            /* 顔認識ライブラリ外部に対するエラー報告
2190               デバッグ、動作確認のため必要不可欠　*/
2191            CString CFaceBase::reportError()
2192            {
2193                    return errorText;       /* エラーを示すグローバル文字列を返
2194    す　*/
2195            }
2196
2197            /* 顔認識エンジンに外部より顔画像を登録するために用いられる
2198               ユースケース記述では「個人情報の検索」の部分で利用する
2199               基本的には顔画像を取り扱う個人情報型（CPERSON)、顔写真型
2200    （CPERSONIMG）
2201               のインターフェース　*/
2202            void CFaceBase::AddImage( long personID, CvvImage* image, bool
2203    loaded )
2204            {
2205
2206                    /* TRUE なら顔認識命令で利用したデータを流用、FALSE な
2207    らこれから読込
2208                       今回のアプリケーションでは前者の機能を利用 */
2209                    CPerson *Person = FindPersonByPersonID( personID );
2210                    if( loaded )
2211                    {
2212                            CvvImage *data = new CvvImage();
2213                            data->CopyOf(     m_trained_image->GetImage()     );
2214            // 計算機に負荷をかけるがこの作業は必要
2215                            Person->AddImage( data, NULL );
2216                    }
2217                    else
2218                            Person->AddImage( image, NULL );
2219                    Person->SetModified();
2220                    SetModified( true );
```

```
2221            }
2222
2223            /* 顔認識エンジンに外部より顔画像を削除するために用いられる */
2224            void    CFaceBase::RemoveImage( long personID, POSITION pos )
2225            {
2226                    CPerson *Person = FindPersonByPersonID( personID );
2227                    Person->m_imgs.RemoveAt( pos );
2228            }
2229
2230            void    CFaceBase::SetImageSize( CSize size )
2231            {
2232                    m_baseImgSize = size;
2233                    SetModified();
2234            }
2235
2236    }
2237    /*************************************************************************
2238    ********************¥
2239    *                                       CPerson class
2240    *   このクラスは顔認識エンジン内部のある個人に関する情報を一括管理する
2241    *   画像テーブルへの入出力顔認識エンジンのパターン認識学習部分を実装
2242    ¥*************************************************************************
2243    ********************/
2244    namespace FACEBASE
2245    {
2246
2247            /* コンストラク */
2248            CPerson::CPerson( CFaceBase* parent )
2249            {
2250                    m_modified = false;
2251                    m_trained   = false;
2252                    m_parent = parent;
2253                    m_hmm = NULL;
2254                    ASSERT( parent != 0 );
2255            }
2256
2257            /* デストラクタ */
```

```
2258        CPerson::~CPerson()
2259        {
2260                Unload();
2261        }
2262
2263        /* 内部の個人情報に対するアクセスを提供 */
2264        void    CPerson::SetName( const CString& name )
2265        {
2266                m_name = name;
2267                SetModified();
2268        }
2269
2270        void   CPerson::SetPersonID( long PersonID )
2271        {
2272                m_PersonID = PersonID;
2273                SetModified();
2274        }
2275
2276
2277  ////////////////////////////////////////////////////////////////////
2278  // 顔認識テーブルへのアクセス
2279  ////////////////////////////////////////////////////////////////////
2280        /* 受け取ったレコード型データ（テーブルに相当）を解析して顔写真
2281  データを獲得し、
2282            顔写真を保管するリスト構造に登録する
2283            このメソッドは膨大な時間を必要とするので、ユースケース記述
2284  「個人情報の検索」で
2285            パターン認識学習を始めるまで利用しない */
2286        HRESULT   CPerson::Load( _RecordsetPtr m_pRecSet_img )
2287        {
2288                /* 必要とする作業用変数 */
2289                CPersonImage *pImage;
2290                CvvImage *image = new CvvImage();
2291                HRESULT hr;
2292                RECT roi;
2293
2294                /* 画像リストのクリア */
```

```
2295                    m_imgs.RemoveAll();
2296                    //errorText.AppendFormat(_T("画像データベース接続直後
2297    ¥n"));
2298
2299                    /* 画像情報テーブルを開く */
2300                    try
2301                    {
2302                            /* 画像情報テーブルを開く */
2303                            m_pRecSet_img->MoveFirst();
2304                            while( !m_pRecSet_img->EndOfFile )
2305                            {
2306                                    /* 顔認識 ID（PERSONID）に該当するデータ
2307    であることの確認
2308                                       この方式は洗練されておらず大変な不効率
2309    を招いている可能性があるので注意 */
2310
2311            if( (long)(m_pRecSet_img->Fields->Item["PERSONID"]->GetValue()
2312    ) != m_PersonID )
2313                                    {
2314                                            //errorText.AppendFormat(_T("%d  "),
2315    m_pRecordSet->Fields->Item["PERSONID"]->GetValue());
2316                                            if( !m_pRecSet_img->EndOfFile )
2317
2318            m_pRecSet_img->MoveNext();
2319                                            continue;
2320                                    }
2321
2322                                    /* 画像データをデータベースより読込 */
2323                                    hr  =  ImageDownLoader(  m_pRecSet_img,
2324    "IMAGE", image );
2325                                    if( FAILED(hr) )
2326                                    {
2327                                            errorText.AppendFormat(_T("画像ダ
2328    ウンロード失敗¥n"));
2329                                            m_pRecSet_img->MoveNext();
2330                                            continue;
2331                                    }
```

```
2332                                    if( !m_pRecSet_img->EndOfFile )
2333                                            m_pRecSet_img->MoveNext();
2334
2335                                    /* 画像データの登録 */
2336                                    pImage = new CPersonImage();
2337                                    pImage->SetModified( false );
2338                    //重要：保存側での効率に大きく影響
2339                                    IplImage *src_data = image->GetImage();
2340                                    pImage->GetImage().CopyOf( src_data );
2341
2342                                    /* 画像データを保管するリスト構造にデータ
2343    の付加 */
2344                                    if(   image == NULL )
2345                                    {
2346                                            delete pImage;
2347
2348            errorText.AppendFormat(_T("ERROR    in    append    image    to
2349    CImagelist¥n"));
2350                                            continue;
2351                                    }
2352                            m_imgs.AddTail( pImage );
2353
2354                            }
2355                    }
2356                    catch( _com_error &e )          /* エラー処理 */
2357                    {
2358                            throw e;
2359                            return hr;
2360                    }
2361
2362                    return S_OK;
2363            }
2364
2365            /* データベースのレコードより画像と画像に関連する情報の読込
2366               画像テーブル読込関数の中核部分を実装 */
2367            HRESULT         CPerson::ImageDownLoader(         _RecordsetPtr
2368    m_pRecordSet, LPCSTR dataROW, CvvImage *newImage )
```

```
2369        {
2370                /* 必要とする作業用変数 */
2371                RECT roi;
2372
2373
2374                /* 画像ファイルの処理用配列の設定 */
2375                _variant_t varChunk;                    // 画像ファイルの処理
2376    用配列の設定
2377                long lngOffSet,lngLogoSize;
2378                lngOffSet = 0;
2379                UCHAR chData;
2380                HRESULT hr = S_OK;
2381
2382                SAFEARRAY FAR *m_pImageData;  // 作業用セーフ配列の
2383    設定
2384                SAFEARRAYBOUND rgsabound[1];
2385                rgsabound[0].lLbound = 0;
2386                rgsabound[0].cElements = CHUNKSIZE;
2387
2388                /* データの画像クラスへの読込 */
2389                CComPtr<IStream> imageStream;
2390                try
2391                {
2392                        /* サイズの取得と空白値対策 */
2393                        lngLogoSize                                             =
2394    m_pRecordSet->Fields->Item[(LPCSTR)dataROW]->ActualSize;
2395                        if( lngLogoSize == 0 )
2396                        {
2397                                return E_FAIL;
2398                        }
2399
2400                        /* データを格納する BYTE 型の SafeArray 配列の設定
2401    */
2402                        /* 画像データはバイナリデータなので読み込んだ時点
2403    では BYTE 型の配列データとみなされる
2404                                SafeArray 配列は配列関係の様々な機能を提供する
2405    クラスライブラリ */
```

```
2406                        rgsabound[0].cElements = lngLogoSize;
2407                        m_pImageData                                    =
2408  SafeArrayCreate(VT_UI1,1,rgsabound);
2409                        if( m_pImageData == NULL )
2410                              return E_OUTOFMEMORY;
2411
2412                        /* GetChunk メソッドを利用した配列データの読み出
2413  し
2414                           データ読込は通信の一種として行われるので、バッ
2415  ファを経由してデータをやりとりする */
2416                        long index1 = 0;
2417                        while(lngOffSet < lngLogoSize)
2418                        {
2419                              varChunk                                  =
2420  m_pRecordSet->Fields->Item[(LPCSTR)dataROW]->GetChunk(CHUNKSI
2421  ZE);
2422
2423                              //Copy the data only upto the Actual Size of
2424  Field.
2425                              for(long
2426  index=0;index<=(CHUNKSIZE-1);index++)
2427                              {
2428                                    hr=
2429  SafeArrayGetElement(varChunk.parray,&index,&chData);
2430                                    if(SUCCEEDED(hr))
2431                                    {
2432                                          //Take BYTE by BYTE and
2433  advance Memory Location
2434                                          hr                        =
2435  SafeArrayPutElement(m_pImageData,&index1,&chData);
2436                                          index1++;
2437                                    }
2438                                    else
2439                                          break;
2440                              }
2441                              lngOffSet = lngOffSet + CHUNKSIZE;
2442                        }
```

```
2443                        lngOffSet = 0;
2444
2445                        /* 獲得した配列データを画像データに変換 */
2446                        //errorText.AppendFormat(_T("size = %d¥n"),
2447 m_pImageData->rgsabound[0].cElements );
2448                        hr = SafeArrayToCvvImage( m_pImageData,
2449 newImage );
2450                        if( FAILED(hr) )
2451                        {
2452                                errorText.AppendFormat(_T("画像ダウンロー
2453 ド最終作業失敗¥n"));
2454                                return hr;
2455                        }
2456                }
2457                catch( _com_error &e )          /* エラー処理 */
2458                {
2459                        throw e;
2460                        return hr;
2461                }
2462
2463                return hr;
2464        }
2465
2466        /* CPERSON クラスの終了処理 */
2467        void  CPerson::Unload()
2468        {
2469                while( !m_imgs.IsEmpty() )
2470                {
2471                        CPersonImage* image = m_imgs.RemoveHead();
2472                        delete image;
2473                }
2474        }
2475
2476        /* 受け取ったレコード型データ（テーブルに相当）に顔写真データを
2477 保存する、
2478              このメソッドもユースケース記述「個人情報の検索」でパターン認
2479 識学習を始めるまで利用しない */
```

```
2480        HRESULT   CPerson::Save( _RecordsetPtr m_pRecordSet )
2481        {
2482            /* 必要とする作業用変数 */
2483            CPersonImage* pImage = NULL;
2484        /* データ付加用の新規インスタンス */
2485            HRESULT hr;
2486
2487            /* 顔写真が新規に付加された場合のみこのメソッドは実質的
2488    に機能 */
2489            if( IsModified() )
2490            {
2491                /* 画像情報の更新 */
2492                try
2493                {
2494                    /* データ処理本体 */
2495                    POSITION pos = m_imgs.GetHeadPosition();
2496                    m_pRecordSet->MoveLast();
2497                    while( pos )
2498                    {
2499                        /* 更新するべきデータを検索 */
2500                        pImage = m_imgs.GetNext( pos );
2501                        if( !pImage->IsModified() )
2502                            continue;
2503
2504                        /* 新レコードの作製 */
2505                        hr = m_pRecordSet->AddNew();
2506                        if( FAILED(hr))
2507                        {
2508                            errorText.AppendFormat(_T("
2509    新規レコード付加失敗¥n"));
2510                            return hr;
2511                        }
2512                        errorText.AppendFormat(_T("画像デ
2513    ータベース更新要求成功¥n"));
2514
2515                        /* データベースへの値の入力 */
2516
```

```
2517            m_pRecordSet->Fields->Item["PERSONID"]->PutValue( m_PersonI
2518    D );
2519                                            hr = ImageUpLoader( m_pRecordSet,
2520    "IMAGE", &(pImage->GetImage()) );
2521                                            pImage->SetModified( false );
2522                                    }
2523
2524                                    /* データベースの更新を実施 */
2525                                    hr = m_pRecordSet->Update();
2526                                    if( FAILED(hr))
2527                                            return CTL_E_DEVICEIOERROR;
2528                                    errorText.AppendFormat(_T("画像データベー
2529    ス更新完了¥n"));
2530                            }
2531                            catch( _com_error &e )          /* エラー処理 */
2532                            {
2533                                    throw e;
2534                                    return hr;
2535                            }
2536
2537                            /* 画像テーブルの更新完了を設定：プログラムの効率
2538    性のため重要 */
2539                            SetModified(false);
2540                    }
2541
2542                    return S_OK;
2543            }
2544
2545            /* データベースのレコードに対して画像と画像に関連する情報の保存
2546                画像テーブル保存メソッドの中核部分を実装 */
2547            HRESULT CPerson::ImageUpLoader( _RecordsetPtr m_pRecordSet,
2548    LPCSTR datarow, CvvImage *newImage )
2549            {
2550                    /* 作業用変数 */
2551                    _variant_t varChunk;                                    // 画像
2552    処理関係
2553                    long lngOffSet;
```

```
2554                    lngOffSet = 0;
2555                    HRESULT hr = S_OK;
2556
2557                    /* 作業用セーフ配列の設定と画像の配列データへの変換 */
2558                    SAFEARRAY                   *m_pImageData                   =
2559    SafeArrayFromCvvImage( newImage );
2560                    if( m_pImageData == NULL )
2561                            return E_FAIL;
2562                    errorText.AppendFormat(_T("size             =             %d¥n"),
2563    m_pImageData->rgsabound[0].cElements );
2564
2565                    /* 画像テーブルへの保存 */
2566                    try
2567                    {
2568                            varChunk.vt = VT_ARRAY|VT_UI1;
2569                            varChunk.parray = m_pImageData;
2570                            hr                                                      =
2571    m_pRecordSet->Fields->Item[datarow]->AppendChunk(varChunk);
2572                    }
2573                    catch( _com_error &e )          /* エラー処理 */
2574                    {
2575                            throw e;
2576                            return hr;
2577                    }
2578
2579                    return hr;
2580            }
2581
2582            void CPerson::DeleteHMMInfo()
2583            {
2584                    m_trained = false;
2585                    ClearHMM();
2586                    m_hmm = NULL;
2587            }
2588
2589            ////////////////////////////////////////////////////////////////////
2590            // CIMAGELIST 型（データ保管用に設定された顔写真データのリスト
```

```
2591  構造）へのアクセスの実装
2592          // リスト構造というのはデータ構造の一種で、オブジェクト指向の言
2593  語では
2594          // データ処理のプログラムごとライブラリとして提供される。
2595          //////////////////////////////////////////////////////////////////
2596          /* 顔写真データのリスト構造（CIMAGELIST）に対して顔写真デー
2597  タを付加 */
2598          /* ユースケース記述「個人情報の検索」に相当する作業で利用 */
2599          void    CPerson::AddImage( CvvImage* import_image, CRect rect )
2600          {
2601                  if( import_image == NULL)
2602                          return;
2603
2604                  /* リストに付加するべき CPersonImage 型インスタンスの作
2605  成 */
2606                  CPersonImage* image;
2607                  image = new CPersonImage();
2608
2609                  /* CVVIMAGE 型データのコピー */
2610                  CvvImage& dst_img = image->GetImage();
2611                  IplImage* src_img = import_image->GetImage();
2612                  ASSERT( src_img != 0 );
2613                  dst_img.CopyOf( *import_image, 0 );
2614
2615                  /* その他の CVVIMAGE 型データの設定 */
2616                  image->SetRoiInFile( CRect( 0, 0, import_image->Width(),
2617  import_image->Height() ) );
2618                  image->SetModified();
2619
2620                  /* リストに付加 */
2621                  m_imgs.AddTail( image );
2622                  SetModified();
2623                  m_trained = false;
2624          }
2625
2626          /* 顔写真データのリスト構造（CIMAGELIST）より指定された顔写
2627  真を削除 */
```

```
2628        void   CPerson::RemoveImage( POSITION pos )
2629        {
2630                CPersonImage* image = m_imgs.GetAt( pos );
2631
2632                if( image )
2633                {
2634                        m_imgs.RemoveAt( pos );
2635                        image->Unload();
2636                        delete image;
2637                        SetModified();
2638                }
2639        }
2640
2641        /* 顔写真データのリスト構造（CIMAGELIST）の先頭に移動 */
2642        void   CPerson::MoveToTop( POSITION pos )
2643        {
2644                CPersonImage* image = m_imgs.GetAt( pos );
2645                if( image )
2646                {
2647                        m_imgs.RemoveAt( pos );
2648                        CPersonImage* old_head = m_imgs.GetHead();
2649                        if( old_head ) old_head->Unload();
2650                        m_imgs.AddHead( image );
2651                        SetModified( true );
2652                }
2653        }
2654
2655        /* 顔画像認識エンジンのパターン認識学習部分の実装
2656            こちらはOPENCVのサンプルとして提供されたコードをそのまま
2657    利用
2658            一応解析し一部は改造したが、基本的には正体不明のプログラム */
2659
2660        /* 画像の輝度補正に関する関数：画像データをパターン認識学習用に
2661    処理する関数 */
2662        void LightingCorrection(IplImage* ipl_image)
2663        {
2664                CvSize roi;
```

```
2665                    int step;
2666                    uchar* img;
2667
2668                    cvGetImageRawData(ipl_image, &img, &step, &roi);
2669
2670                    int i, j;
2671                    int width = roi.width;
2672                    int height = roi.height;
2673
2674                    float x1, x2, y1, y2;
2675                    int f[3] = {0, 0, 0};
2676                    float a[3] = {0, 0, 0};
2677
2678                    float h1;
2679                    float h2;
2680
2681                    float c1,c2;
2682
2683                    float min = FLT_MAX;
2684                    float max = -FLT_MAX;
2685                    float correction;
2686
2687                    float* float_img = (float*)malloc( width * height *
2688  sizeof(float) );
2689
2690                    x1 = width * (width + 1) / 2.0f; // Sum (1, ... , width)
2691                    x2 = width * (width + 1 ) * (2 * width + 1) / 6.0f; // Sum
2692  (1^2, ... , width^2)
2693                    y1 = height * (height + 1)/2.0f; // Sum (1, ... , width)
2694                    y2 = height * (height + 1 ) * (2 * height + 1) / 6.0f; // Sum
2695  (1^2, ... , width^2)
2696
2697
2698                    // extract grayvalues
2699                    for (i = 0; i < height; i++)
2700                    {
2701                            for (j = 0; j < width; j++)
```

```
2702                    {
2703                            f[2] = f[2] + j * img[i*step + j];
2704                            f[1] = f[1] + i * img[i*step + j];
2705                            f[0] = f[0] +       img[i*step + j];
2706                    }
2707            }
2708
2709            h1 = (float)f[0] * (float)x1 / (float)width;
2710            h2 = (float)f[0] * (float)y1 / (float)height;
2711
2712            a[2]    =    ((float)f[2]    -    h1)    /    (float)(x2*height    -
2713    x1*x1*height/(float)width);
2714            a[1]    =    ((float)f[1]    -    h2)    /    (float)(y2*width    -
2715    y1*y1*width/(float)height);
2716            a[0]        =        (float)f[0]/(float)(width*height)        -
2717    (float)y1*a[1]/(float)height -
2718                    (float)x1*a[2]/(float)width;
2719
2720            for (i = 0; i < height; i++)
2721            {
2722                    for (j = 0; j < width; j++)
2723                    {
2724
2725                            correction = a[0] + a[1]*(float)i + a[2]*(float)j;
2726
2727                            float_img[i*width + j] = img[i*step + j] -
2728    correction;
2729
2730                            if (float_img[i*width + j] < min) min =
2731    float_img[i*width+j];
2732                            if (float_img[i*width + j] > max) max =
2733    float_img[i*width+j];
2734                    }
2735            }
2736
2737            //rescaling to the range 0:255
2738            c2 = 0;
```

```
2739                    if (max == min)
2740                            c2 = 255.0f;
2741                    else
2742                            c2 = 255.0f/(float)(max - min);
2743
2744                    c1 = (-(float)min)*c2;
2745
2746                    for (i = 0; i < height; i++)
2747                    {
2748                            for (j = 0; j < width; j++)
2749                            {
2750                                    int value = (int)floor(c2*float_img[i*width +
2751    j] + c1);
2752                                    if (value < 0) value = 0;
2753                                    if (value > 255) value = 255;
2754                                    img[i*step + j] = (uchar)value;
2755                            }
2756                    }
2757
2758                    free( float_img );
2759
2760            }
2761
2762            /* HMM（隠れマルコフモデル）を利用した顔情報の認識トレーニング
2763    本体 */
2764            void CPerson::TrainHMM()
2765            {
2766                    int color[24] = { RGB(255,128,128), RGB(255,255,128),
2767    RGB(128,255,128), RGB(128,255,255), RGB(0,128,255),
2768                                    RGB(255,128,255), RGB(255,0,0),
2769    RGB(255,128,0), RGB(0,128,0), RGB(0,0,0), RGB(255,255,128),
2770                                    RGB(255,0,128), RGB(255,128,128),
2771    RGB(255,255,128),RGB(128,255,128), RGB(128,255,255),
2772
2773    RGB(0,128,255),RGB(255,128,255),RGB(255,0,0),RGB(255,128,0),RGB(0,12
2774    8,0),
2775                                    RGB(0,0,0),RGB(255,255,128),
```

```
2776    RGB(255,0,128)   };
2777
2778                    //認識ループが収束しない場合の設定
2779                    const int max_iterations = 80;
2780
2781                    //アクセス元となる顔画像データベースの設定
2782                    CFaceBase* parent = GetParentBase();
2783                    if( parent == NULL )                                    // 空 ポ
2784    インタを入力した場合の対策
2785                            return;
2786                    //CImage& segmentation = parent->GetTrainedImage();
2787                    //segmentation.Create( 320, 320, 24 );
2788
2789                    //画像ファイルのサイズの設定（最初に計算して得られた DCT
2790    係数の圧縮）
2791                    int     vect_len    =    parent->m_obsSize.height    *
2792    parent->m_obsSize.width;
2793                    if( parent->m_suppress_intensity )
2794                    {
2795                            vect_len--;
2796                    }
2797
2798                    //HMM 情報を登録する変数の確保
2799                    CvEHMM* hmm = m_hmm->GetIppiEHMM();
2800                    if  (!hmm)  m_hmm->CreateHMM(  parent->m_stnum,
2801    parent->m_mixnum, vect_len );
2802                    hmm = m_hmm->GetIppiEHMM();
2803
2804                    //認識トレーニングに用いる画像ファイルへのリストの設定
2805                    int num_img_real = m_imgs.GetCount();
2806                    int num_img = 0;
2807                    if( num_img_real > LIMIT_RECOGNIZE )
2808                            num_img = LIMIT_RECOGNIZE;
2809                    else
2810                            num_img = num_img_real;
2811                    CAtlArray< CvImgObsInfo* > obs_info;
2812                    obs_info.SetCount( num_img );       //データ格納用配列の
```

```
サイズ決定
                CvImgObsInfo** obs_info_array = obs_info.GetData();

                //認識トレーニングに用いる画像に対する設定と処理
                CPersonImage *personImg;
                POSITION pos = m_imgs.GetTailPosition();
                for( int i = 0; i < num_img; i++ )
                {
                        //認識に利用する画像の選択（最大画像数は
LIMIT_RECOGNIZE で設定）
                        //POSITION pos = m_imgs.FindIndex(i);
                        //IplImage*                ipl                =
m_imgs.GetAt(pos)->GetImage().GetImage();
                        CPersonImage *personImg = m_imgs.GetPrev( pos );
                        IplImage* ipl = personImg->GetImage().GetImage();

                        //このプログラムで再描画に利用する画像サイズの計
算
                        bool doRescale = false;
                        int new_height = 0;
                        int new_width = 0;

                        if ( parent->m_useWidth )
                        {
                                doRescale = true;
                                new_width = parent->m_scaleWidth;
                        }
                        if ( parent->m_useHeight )
                        {
                                doRescale = true;
                                new_height = parent->m_scaleHeight;
                        }

                       //errorText.AppendFormat(_T("new_height = %d¥n"),
new_height );
                        //errorText.AppendFormat(_T("new_width = %d¥n"),
new_width );
```

```
2850                        //errorText.AppendFormat(_T("Image Height
2851    = %d¥n"), m_imgs.GetAt(pos)->GetImage().Height());
2852                        //continue;//imageData->GetHeight()
2853
2854                        //再描画メッセージに対する処理
2855                        IplImage* ipl_scaled;
2856                        CvSize image_roi = cvSize( ipl->roi ? ipl->roi->width :
2857    ipl->width,
2858                                ipl->roi ? ipl->roi->height :
2859    ipl->height );
2860                        if ( doRescale )
2861                        {
2862                                //再描画に必要な計算処理
2863                                if ( !new_width )
2864                                {
2865                                        new_width = new_height *
2866    image_roi.width / image_roi.height;
2867                                }
2868                                else if ( !new_height )
2869                                {
2870                                        new_height = new_width *
2871    image_roi.height / image_roi.width;
2872                                }
2873
2874                                //再描画の実行
2875                                ipl_scaled =
2876    cvCreateImage( cvSize(new_width, new_height), IPL_DEPTH_8U, 1 );
2877
2878                                cvResize(ipl, ipl_scaled, /*ipl_scaled->width,
2879    ipl->width,
2880                                 ipl_scaled->height, ipl->height,*/ CV_INTER_NN);
2881                        }
2882                        else
2883                                ipl_scaled = ipl;
2884
2885                        //このプログラムで以降の計算に利用する画像サイズ
2886    の計算と設定
```

```
2887                        CvSize roi = cvSize( ipl_scaled->roi ?
2888    ipl_scaled->roi->width : ipl_scaled->width,
2889                                ipl_scaled->roi ? ipl_scaled->roi->height :
2890    ipl_scaled->height);
2891                        CvSize num_obs;
2892                        CV_COUNT_OBS( &roi, &(parent->m_dctSize),
2893    &(parent->m_delta), &num_obs );
2894                        obs_info_array[i] = cvCreateObsInfo( num_obs,
2895    vect_len );
2896                        CvImgObsInfo* info = obs_info_array[i];
2897
2898                        //IplImage* normalized_image = cvCreateImage( roi,
2899    IPL_DEPTH_8U, 1 );
2900                        //NormalizeImageForHMM( ipl_scaled,
2901    normalized_image );
2902
2903                        //ファイルが圧縮された場合の処理
2904                        if( parent->m_suppress_intensity )
2905                        {
2906                            float* observations =
2907    (float*)malloc( num_obs.height * num_obs.width * (vect_len+1) *
2908    sizeof(float) );
2909
2910            cvImgToObs_DCT( /*normalized_image*/ipl_scaled, observations,
2911    parent->m_dctSize, parent->m_obsSize, parent->m_delta );
2912                            ExtractDCT( observations, info->obs,
2913    num_obs.height * num_obs.width, vect_len );
2914                            free( observations);
2915                        }
2916                        else
2917                        {
2918
2919            cvImgToObs_DCT( /*normalized_image*/ipl_scaled, info->obs,
2920    parent->m_dctSize, parent->m_obsSize, parent->m_delta );
2921                        }
2922                        //ファイルが圧縮された場合に備え再描画処理
2923                        if ( doRescale )
```

```
2924                    {
2925                            cvReleaseImage( &ipl_scaled );
2926                    }
2927
2928                    //cvReleaseImage( &normalized_image );
2929
2930                    cvUniformImgSegm( info, hmm );
2931
2932            }
2933
2934            //HMM トレーニングに必要な初期の設定
2935            cvInitMixSegm( obs_info_array, num_img, hmm );
2936            bool trained = false;
2937            float old_likelihood = 0;
2938            int counter = 0;
2939
2940            //認識動作の実行
2941            while( (!trained) && (counter < max_iterations) )
2942            {
2943                    counter++;
2944
2945                    int j;
2946    #if 0
2947                    //segment images
2948                    for( j = 0; j < 1; j++ )
2949                    {
2950                            IplImage*               ipl_segm                =
2951    segmentation.GetImage();
2952                            CvImgObsInfo* obs = obs_info_array[j];
2953
2954                            int counter=0;
2955                            for(int k = 0; k < obs->obs_y; k++ )
2956                            {
2957                                    for(int   m   =   0;   m   <   obs->obs_x;
2958    m++,counter++ )
2959                                    {
2960                                            cvCircle(               ipl_segm,
```

```
2961    cvPoint( (parent->m_dctSize.width>>1) +
2962
2963                            (parent->m_delta.width)* m ,
2964
2965                            (parent->m_dctSize.height>>1) +
2966
2967                            (parent->m_delta.height)* k ), 3,
2968
2969                             color[obs->state[counter*2+1]], 1 );
2970                                        }
2971                                }
2972
2973                                parent->SetTrainedIndex(j);
2974                                parent->UpdateTrainedImage();
2975
2976                            }
2977    #endif
2978                            // 隠れマルコフモデルのパラメータを計算する：
2979    OPENCV のライブラリ関数を利用
2980                            cvEstimateHMMStateParams(       obs_info_array,
2981    num_img, hmm);
2982                            cvEstimateTransProb(   obs_info_array,   num_img,
2983    hmm);
2984
2985                            float likelihood = 0;
2986                            for( j = 0; j < num_img; j++ )
2987                            {
2988                                cvEstimateObsProb(       obs_info_array[j],
2989    hmm );
2990                                likelihood  +=  cvEViterbi(  obs_info_array[j],
2991    hmm );
2992                            }
2993                            likelihood /= num_img*obs_info_array[0]->obs_size;
2994
2995                            cvMixSegmL2( obs_info_array, num_img, hmm);
2996
2997                            trained = ( fabs(likelihood - old_likelihood) < 0.01 );
```

```
2998                        old_likelihood = likelihood;
2999                }
3000
3001                //画像に利用したメモリ上の空間を解放
3002                for(i = 0; i < num_img; i++ )
3003                {
3004                        cvReleaseObsInfo( &(obs_info_array[i]) );
3005                }
3006
3007                obs_info.RemoveAll();
3008                //    segmentation.Destroy();
3009
3010                //モードをトレーニング済みに戻して作業終了
3011                m_trained = true;
3012
3013                errorText.AppendFormat(_T("トレーニング終了¥n"));
3014        }
3015
3016  }
3017
3018  /*************************************************************************
3019  ********************¥
3020  *                                    CPersonImage class
3021  *   このクラスは本ライブラリで利用する顔写真に関する情報を一括管理する
3022  *   本ライブラリで必要なデータベース関係のうちの一部を担当する*
3023  ¥*************************************************************************
3024  *********************/
3025  namespace FACEBASE
3026  {
3027        /* コンストラクタ */
3028        CPersonImage::CPersonImage()
3029        {
3030                m_modified = false;
3031                m_roi_in_file = CRect(0,0,0,0);
3032        }
3033
3034        /* デストラクタ */
```

```
3035        CPersonImage::~CPersonImage()
3036        {
3037                Unload();
3038        }
3039
3040        void   CPersonImage::Unload()
3041        {
3042                m_img.Destroy();
3043        }
3044
3045        void   CPersonImage::SetRoiInFile( CRect r )
3046        {
3047                m_roi_in_file = r;
3048                SetModified( m_img.Width() != 0 );
3049        }
3050
3051
3052        /*************************************************************************
3053 ***************************¥
3054        *                                        その他極めて重要な実装(1)
3055        *   認識トレーニングメソッドへのインターフェースを実装
3056        ¥*************************************************************************
3057 ****************************/
3058        /* 指定された人間（KEY）に対して顔認識トレーニングを実施
3059            ユースケース記述「個人情報の検索」で行う顔認識のインターフェ
3060 ース  */
3061        void CFaceBase::TrainPerson( long PersonID )
3062        {
3063                CPerson* person = FindPersonByPersonID( PersonID );
3064                ASSERT( person );
3065                if (!person) return;
3066                if (person->IsTrained() ) return;
3067                CPersonImgList& imgs = person->GetImgList();
3068                if( imgs.GetCount() == 0 ) return;
3069
3070                // 画像データの読込
3071                image_load = true;
```

```
3072                    Load();
3073                    image_load = false;
3074
3075                    // 指定された人間（KEY）に対して顔認識トレーニングを実施
3076                    person->TrainHMM();
3077                    person->SetModified();
3078                    SetModified();
3079            }
3080
3081            /* 条件に叶った全ての人間に対して顔認識トレーニングを実施 */
3082            void CFaceBase::TrainAll()
3083            {
3084                    int end = 0;
3085
3086                    // 更新するべき個人のピックアップ
3087                    int *member = (int *)malloc( m_base.GetCount() *
3088    sizeof(int) );
3089                    POSITION pos = m_base.GetHeadPosition();
3090                    while( pos )
3091                    {
3092                            CPerson* person = m_base.GetNext(pos);
3093                            if ( !person->m_trained )
3094                            {
3095                                    *( member + end ) = person->GetPersonID();
3096                                    end++;
3097                            }
3098                    }
3099
3100                    // 画像データの読込
3101                    image_load = true;
3102                    Load();
3103                    image_load = false;
3104
3105                    // 条件に一致する個人全員へのトレーニング
3106                    for( int j = 0; j < end; j++ )
3107                    {
3108                            CPerson*                person                =
```

```
3109    FindPersonByPersonID(   *( member + j ) );
3110                            errorText.AppendFormat(_T("target     =     %d¥n"),
3111    person->GetPersonID() );
3112                            person->TrainHMM();
3113                    }
3114            }
3115
3116            /*****************************************************************
3117    ****************************¥
3118            *                                       その他極めて重要な実装(2)
3119            *   顔認識エンジンの顔認識プログラムの実装
3120            ¥*****************************************************************
3121    ****************************/
3122            /* 顔認識エンジンの本体、ユースケース記述「個人の識別」で利用    */
3123            /* 認識結果は顔認識テーブルの主キーと一致率（数値の意味は不明）
3124    で返される（key、likelihood)。    */
3125            /* 更に、指定したキーを除外して再検索を実施することも可能である
3126     （remove_key)。    */
3127            int  CFaceBase::RecognizePerson(  CvvImage  *image,  CRect  rect,
3128    long *key, long *remove_key, float *likelihood )
3129            {
3130                    /* ウィンドウサイズの登録内容の調整 */
3131                    if ( rect == CRect( 0,0,0,0 ) )
3132                    {
3133                            rect.bottom = image->Height();
3134                            rect.right = image->Width();
3135                    }
3136
3137                    /* 個人名リスト関係の操作    */
3138                    float   like_array[1000];   ASSERT(   m_base.GetCount()   <
3139    1000 );
3140                    int three_first[REGNZ_NUMBER];
3141
3142                    /* 認識用画像の作成 */
3143                    IplImage* ipl;
3144                    if( image == NULL )
3145                  ipl = m_trained_image->GetImage();
```

```
3146                    else
3147                            ipl = image->GetImage();
3148                    cvSetImageROI( ipl, RectToCvRect( rect ) );
3149                    int code = 1;
3150
3151                    /* プログラム上で利用する画像サイズの設定　*/
3152            bool doRescale = false;
3153            int new_height = 0;
3154            int new_width = 0;
3155             if ( m_useWidth )
3156            {
3157                 doRescale = true;
3158                 new_width = m_scaleWidth;
3159            }
3160            if ( m_useHeight )
3161            {
3162                 doRescale = true;
3163                 new_height = m_scaleHeight;
3164            }
3165
3166            //再描画メッセージに対応する処理
3167            IplImage* ipl_scaled;
3168            if ( doRescale )
3169            {
3170                            //再描画用の画像サイズ計算
3171                 if ( !new_width )
3172                 {
3173                      new_width       =    new_height    *    ipl->roi->width    /
3174    ipl->roi->height;
3175                 }
3176                 else if ( !new_height )
3177                 {
3178                      new_height       =    new_width    *    ipl->roi->height    /
3179    ipl->roi->width;
3180                 }
3181
3182                 //再描画
```

```
3183                ipl_scaled = cvCreateImage( cvSize( new_width, new_height ),
3184    IPL_DEPTH_8U, 1 );
3185
3186                cvResize(ipl, ipl_scaled, /*ipl_scaled->width, ipl->width,
3187                            ipl_scaled->height, ipl->height,*/ CV_INTER_NN);
3188
3189            }
3190            else
3191                ipl_scaled = ipl;
3192
3193                //圧縮画像、および通常画像へ対応するデータ処理
3194                CvSize cvroi = cvSize( ipl_scaled->roi ?
3195    ipl_scaled->roi->width : ipl_scaled->width,
3196
3197            ipl_scaled->roi ? ipl_scaled->roi->height : ipl_scaled->height);
3198                CvSize num_obs;
3199                CvImgObsInfo* info;
3200
3201                CV_COUNT_OBS( &cvroi, &m_dctSize, &m_delta,
3202    &num_obs );
3203
3204                int vect_len = m_obsSize.height*m_obsSize.width;
3205                if( m_suppress_intensity )
3206                {
3207                        vect_len--;
3208                }
3209                info = cvCreateObsInfo( num_obs, vect_len );
3210                if( m_suppress_intensity )
3211                {
3212                        float* observations = (float*)malloc( num_obs.height
3213    * num_obs.width * (vect_len+1) * sizeof(float) );
3214                        cvImgToObs_DCT( /*normalized_image*/ipl_scaled,
3215    observations, m_dctSize, m_obsSize, m_delta );
3216                        ExtractDCT( observations, info->obs,
3217    num_obs.height * num_obs.width, vect_len );
3218                        free( observations);
3219                }
```

```
3220                    else
3221                    {
3222
3223                            //IplImage*             normalized_image            =
3224    cvCreateImage( cvroi, IPL_DEPTH_8U, 1 );
3225                            //NormalizeImageForHMM(                 ipl_scaled,
3226    normalized_image );
3227
3228                            cvImgToObs_DCT(   /*normalized_image*/ipl_scaled,
3229    info->obs, m_dctSize, m_obsSize, m_delta );
3230
3231                    //cvReleaseImage(&normalized_image);
3232                    }
3233                    if ( doRescale )
3234                    {
3235                            cvReleaseImage( &ipl_scaled );
3236                    }
3237
3238                    //各個人（の画像）の読込、HMM 近似度の計算、リストへの
3239    登録
3240                    float max_like = -100000000;
3241                    //int recognized = -1;
3242                    for( int i = 0 ; i < m_base.GetCount(); i++ )
3243                    {
3244                            CPerson*                person                  =
3245    m_base.GetAt( m_base.FindIndex(i) );
3246                            CvEHMM* hmm = 0;
3247
3248                            if( !person->IsTrained() )      //認識トレーニングが
3249    されていない場合の処理
3250                            {
3251                                    code = 0;
3252                                    errorText.AppendFormat(_T("IS    BROKEN
3253    認識通過 4-5¥n"));
3254                                    break;
3255                            }
3256
```

```
3257                        //各個人（の画像）に対する HMM モデル情報の取得
3258                        hmm = person->GetHMM().GetIppiEHMM();
3259                        if (!hmm)                                   // 認 識
3260    トレーニングがされていない場合の処理
3261                        {
3262                                code = 0;
3263                                errorText.AppendFormat(_T("IS     BROKEN
3264    認識通過 4-5¥n"));
3265                                break;
3266                        }
3267
3268                        //各個人（の画像）に対する HMM 近似度の計算と登録
3269                        cvEstimateObsProb( info, hmm );
3270                        like_array[i] = cvEViterbi( info, hmm );
3271                }
3272
3273                //画像データの破棄とメモリ解放
3274                cvReleaseObsInfo( &info );
3275                if( !code ) return code;
3276
3277        #define REGNZ_NUMBER 3
3278
3279                //HMM 近似度が最大であるインデックスの報告
3280                CPerson* person;
3281                int k = 0, remove_number = 0;
3282                if( remove_key != NULL )
3283                {
3284                        for(  remove_number  =  0;  *(  remove_key  +
3285    remove_number ) != -1; remove_number++ );
3286                        errorText.AppendFormat(_T("remove_number
3287    = %d¥n"), remove_number );
3288                }
3289                //remove_number = 0;
3290
3291                for( i = 0; i < MIN( REGNZ_NUMBER, m_base.GetCount()) ;
3292    i++ )
3293                {
```

```
3294                    float maxl = -FLT_MAX;
3295                    int maxind = -1;
3296            //その時の最大インデックス
3297
3298                    for( int j = 0; j < m_base.GetCount(); j++ )
3299                    {
3300                            if (like_array[j] > maxl)
3301                            {
3302                                    //除去候補に当たった場合の処理
3303                                    person = GetPerson( j );
3304                                    if( remove_number != 0 )
3305                                    {
3306                                            for(  int  m  =  0;  m  <
3307  remove_number; m++ )
3308                                            {
3309
3310          if( person->GetPersonID() == *( remove_key + m ) )
3311                                                            break;
3312                                            }
3313                                            if( m != remove_number )
3314                                                    continue;
3315                                    }
3316
3317                                    //最大候補の登録
3318                                    maxl = like_array[j];
3319                                    maxind = j;
3320                            }
3321                    }
3322
3323                    three_first[i] = maxind;
3324                    likelihood[i] = like_array[maxind];     //インデックス
3325  に対応する確率の保存
3326                    like_array[maxind] = -FLT_MAX;
3327            }
3328
3329            //HMM 近似度が最大である主キーに変換して報告
3330            for( i = 0; i < REGNZ_NUMBER; i++ )
```

```
3331                {
3332                        person = GetPerson( three_first[i] );
3333                        key[i] = person->GetPersonID();
3334                }
3335
3336                return MIN( REGNZ_NUMBER, m_base.GetCount());
3337        }
3338
3339        /*****************************************************************
****************************¥
3341        *                                       その他の実装
3342        ¥*****************************************************************
****************************/
3344        /* 顔認識データベースのオプションを設定する   */
3345        int CFaceBase::SetParams(    //sampling params
3346                                CvSize dctSize,
3347                                CvSize obsSize,
3348                                CvSize delta,
3349                                //HMM params
3350                                int* states, int mix,
3351                                //image scaling params
3352                                BOOL use_width, int width,
3353                                BOOL use_height, int height,
3354                                BOOL          suppress_intens,          BOOL
3355 leave_hmm_alive)
3356        {
3357                BOOL clearHMM = FALSE;
3358
3359                //サンプリングパラメータの設定
3360                if ( (dctSize.height != m_dctSize.height) || (dctSize.width !=
3361 m_dctSize.width) ||
3362                        (obsSize.height      !=      m_obsSize.height)      ||
3363 (obsSize.width != m_obsSize.width) ||
3364                        (delta.height  !=  m_delta.height)  ||  (delta.width  !=
3365 m_delta.width) ||
3366                        (m_suppress_intensity != suppress_intens ) )
3367                {
```

```
3368                    clearHMM    = TRUE;
3369            }
3370
3371            m_dctSize =    dctSize;
3372            m_obsSize =    obsSize;
3373            m_delta     =    delta;
3374            m_suppress_intensity = suppress_intens;
3375
3376            //HMM（隠れマルコフモデル）パラメータの設定
3377            BOOL dochange = FALSE;
3378
3379            for( int i = 0; i <= states[0]; i++ )
3380            {
3381                    if ( m_stnum[i] != (states[i]) )
3382                    {
3383                            dochange = TRUE;
3384                            break;
3385                    }
3386            }
3387            if ( mix != m_mixnum[0] )
3388                    dochange = TRUE;
3389
3390            if ( dochange )
3391            {
3392                    //HMM（隠れマルコフモデル）パラメータの変更
3393                    int counter = 0;
3394                    for( int i = 0; i <= states[0]; i++ )
3395                    {
3396                            m_stnum[i] = (states[i]);
3397                            for( int j    = 0; j < states[i]; j++, counter++ )
3398                            {
3399                                    m_mixnum[counter]    = mix;
3400                            }
3401                    }
3402                    clearHMM |= dochange;
3403            }
3404
```

```
3405                    //画像サイズのパラメータ設定
3406                    if  (  (m_useWidth  !=  use_width)  ||  (m_useHeight  !=
3407    use_height) )
3408                    {
3409                            clearHMM = TRUE;
3410                    }
3411                    else if ( ( m_useWidth &&   (width != m_scaleWidth) ) ||
3412                    ( m_useHeight && (height != m_scaleHeight) ) )
3413                    {
3414                            clearHMM = TRUE;
3415                    }
3416
3417                    m_useWidth = use_width;
3418                    m_useHeight = use_height;
3419                    m_scaleWidth = width;
3420                    m_scaleHeight = height;
3421
3422                    if( clearHMM && (!leave_hmm_alive) )
3423                    {
3424                            for ( int i = 0; i < m_base.GetCount(); i++ )
3425                            {
3426                                    CPerson* person = GetPerson( i );
3427                                    person->ClearHMM();
3428                                    person->SetModified(true);
3429                            }
3430                    }
3431                    return clearHMM;
3432            }
3433
3434            /* HMM パラメータ部分を無効化する：学習をやり直す   */
3435            void CPerson::ClearHMM()
3436            {
3437                    m_hmm->Release();
3438                    m_trained = false;
3439            }
3440
3441            /* HMM パラメータを返す   */
```

```
3442            CContEHMM& CPerson::GetHMM()
3443            {
3444                    return *m_hmm;
3445            }
3446
3447    }
3448
3449    /*M///////////////////////////////////////////////////////////////////////////////////
3450    //
3451    //    IMPORTANT: READ BEFORE DOWNLOADING, COPYING,
3452    INSTALLING OR USING.
3453    //
3454    //  By downloading, copying, installing or using the software you agree to
3455    this license.
3456    //  If you do not agree to this license, do not download, install,
3457    //  copy or use the software.
3458    //
3459    //
3460    //                          Intel License Agreement
3461    //                  For Open Source Computer Vision Library
3462    //
3463    // Copyright (C) 2000, Intel Corporation, all rights reserved.
3464    // Third party copyrights are property of their respective owners.
3465    //
3466    // Redistribution and use in source and binary forms, with or without
3467    modification,
3468    // are permitted provided that the following conditions are met:
3469    //
3470    //    * Redistribution's of source code must retain the above copyright notice,
3471    //        this list of conditions and the following disclaimer.
3472    //
3473    //    * Redistribution's in binary form must reproduce the above copyright
3474    notice,
3475    //          this list of conditions and the following disclaimer in the
3476    documentation
3477    //        and/or other materials provided with the distribution.
3478    //
```

```
//     * The name of Intel Corporation may not be used to endorse or promote
products
//       derived from this software without specific prior written permission.
//
// This software is provided by the copyright holders and contributors "as is"
and
// any express or implied warranties, including, but not limited to, the
implied
// warranties of merchantability and fitness for a particular purpose are
disclaimed.
// In no event shall the Intel Corporation or contributors be liable for any
direct,
// indirect, incidental, special, exemplary, or consequential damages
// (including, but not limited to, procurement of substitute goods or services;
// loss of use, data, or profits; or business interruption) however caused
// and on any theory of liability, whether in contract, strict liability,
// or tort (including negligence or otherwise) arising in any way out of
// the use of this software, even if advised of the possibility of such damage.
//
//   （補足条項）
//   このプログラムは Intel License Agreement に基づく Intel 社のオープンソ
ースのプログラムである。
//   私、飯塚岳郎が卒業研究の目的でいくつかの重要な改変を行った。
//     （１）ATL に基づく COM コンポーネントへの改装
//     （２）ファイルアクセスのデータベース化
//     （３）IPL 導入による BMP<->IPL（OPENCV）間のフォーマット変換機
能の導入
//
//   このプログラムも Intel License Agreement に準じて公開する。
//   なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負
うものとする
//
//   （本ファイルの内容）ContEHMM.cpp
//   OPENCV 特有の HMM（隠れマルコフモデル）データの管理、及び入出力
クラスのインターフェースの実装
//   （CContEHMM クラスのインターフェース）
//
```

```
3516  //  （本ライブラリに必要な DLL ファイル）
3517  //  highgui.dll, cv.dll, cvaux.dll(OPENCV)
3518  //  IPL.dll, IPLa6.dll(IPL)
3519  //
3520  ////////////////////////////////////////////////////////////////////*/
3521
3522  /* インクルードガード */
3523  #if !defined(AFX_CONTEHMM_H__A078F09B_40E3_421A_8422_C420970
3524  7657A__INCLUDED_)
3525  #define
3526  AFX_CONTEHMM_H__A078F09B_40E3_421A_8422_C4209707657A__IN
3527  CLUDED_
3528
3529  #if _MSC_VER > 1000
3530  #pragma once
3531  #endif // _MSC_VER > 1000
3532  #include "stdafx2.h"
3533
3534  namespace FACEBASE
3535  {
3536          class CContEHMM
3537          {
3538          public:
3539                  /* コンストラクタとデストラクタ */
3540                  BOOL Release();
3541                  CContEHMM();
3542                  virtual ~CContEHMM();
3543                  bool CreateHMM( int* num_states, int* num_mix, int
3544  vect_size );
3545
3546                  /* 顔認識エンジンに対して必要なデータを提供 */
3547                  //IppiEHMM* GetIppiEHMM() { return m_hmm; };
3548                  CvEHMM* GetIppiEHMM() { return m_hmm; };
3549                  int GetVectSize() { return m_vectSize; };
3550
3551                  /* データの入出力（文字列との変換）を行うメソッド */
3552                  bool Save( CString *source );
```

```
3553                    bool Load( CString source );
3554            public:
3555                    CString errorText;      /* エラー文字列 */
3556
3557            protected:
3558                    //IppiEHMM* m_hmm;
3559                    CvEHMM* m_hmm;          /* OPENCV 特有の HMM（隠
3560    れマルコフモデル）データ */
3561                    int m_vectSize;
3562
3563            };
3564    }
3565
3566    #endif
3567    // !defined(AFX_CONTEHMM_H__A078F09B_40E3_421A_8422_C4209707
3568    657A__INCLUDED_)
3569
3570    /*M///////////////////////////////////////////////////////////////////////////////////
3571    //
3572    //    IMPORTANT: READ BEFORE DOWNLOADING, COPYING,
3573    INSTALLING OR USING.
3574    //
3575    //  By downloading, copying, installing or using the software you agree to
3576    this license.
3577    //  If you do not agree to this license, do not download, install,
3578    //  copy or use the software.
3579    //
3580    //
3581    //                          Intel License Agreement
3582    //                  For Open Source Computer Vision Library
3583    //
3584    // Copyright (C) 2000, Intel Corporation, all rights reserved.
3585    // Third party copyrights are property of their respective owners.
3586    //
3587    // Redistribution and use in source and binary forms, with or without
3588    modification,
3589    // are permitted provided that the following conditions are met:
```

```
3590    //
3591    //    * Redistribution's of source code must retain the above copyright notice,
3592    //        this list of conditions and the following disclaimer.
3593    //
3594    //    * Redistribution's in binary form must reproduce the above copyright
3595    notice,
3596    //          this list of conditions and the following disclaimer in the
3597    documentation
3598    //        and/or other materials provided with the distribution.
3599    //
3600    //    * The name of Intel Corporation may not be used to endorse or promote
3601    products
3602    //        derived from this software without specific prior written permission.
3603    //
3604    // This software is provided by the copyright holders and contributors "as is"
3605    and
3606    // any express or implied warranties, including, but not limited to, the
3607    implied
3608    // warranties of merchantability and fitness for a particular purpose are
3609    disclaimed.
3610    // In no event shall the Intel Corporation or contributors be liable for any
3611    direct,
3612    // indirect, incidental, special, exemplary, or consequential damages
3613    // (including, but not limited to, procurement of substitute goods or services;
3614    // loss of use, data, or profits; or business interruption) however caused
3615    // and on any theory of liability, whether in contract, strict liability,
3616    // or tort (including negligence or otherwise) arising in any way out of
3617    // the use of this software, even if advised of the possibility of such damage.
3618    //
3619    //   （補足条項）
3620    //   このプログラムは Intel License Agreement に基づく Intel 社のオープンソ
3621    ースのプログラムである。
3622    //   私、飯塚岳郎が卒業研究の目的でいくつかの重要な改変を行った。
3623    //     （１）ATL に基づく COM コンポーネントへの改装
3624    //     （２）ファイルアクセスのデータベース化
3625    //
3626    //   このプログラムも Intel License Agreement に準じて公開する。
```

```
3627  //   なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負
3628  うものとする
3629  //
3630  //   （本ファイルの内容）ContEHMM.h
3631  //   OPENCV 特有の HMM（隠れマルコフモデル）データの管理、及び入出力
3632  クラスの実装
3633  //   （ContEHMM クラスの実装）
3634  //
3635  //   （本ライブラリに必要な DLL ファイル）
3636  //   highgui.dll, cv.dll, cvaux.dll(OPENCV)
3637  //   IPL.dll, IPLa6.dll(IPL)
3638  //
3639  ////////////////////////////////////////////////////////////////////////*/
3640
3641  /* プログラムに必要なインクルードファイル */
3642  #include "stdafx2.h"
3643  #include "ContEHMM.h"
3644  //#include <stdio.h>
3645  //#include <assert.h>
3646
3647  #ifdef _DEBUG
3648  #undef THIS_FILE
3649  static char THIS_FILE[]=__FILE__;
3650  #define new DEBUG_NEW
3651  #endif
3652
3653  namespace FACEBASE
3654  {
3655          //////////////////////////////////////////////////////////////////////
3656          // Construction/Destruction
3657          //////////////////////////////////////////////////////////////////////
3658          CContEHMM::CContEHMM()
3659          {
3660                  m_hmm = NULL;
3661                  m_vectSize = 0;
3662          }
3663
```

```
3664        CContEHMM::~CContEHMM()
3665        {
3666            if (m_hmm) cvRelease2DHMM( &m_hmm );
3667            m_vectSize = 0;
3668
3669        }
3670
3671        bool CContEHMM::CreateHMM( int* num_states, int* num_mix,
3672    int vect_size )
3673        {
3674            if (m_hmm) cvRelease2DHMM( &m_hmm );
3675            m_hmm = 0;
3676
3677            m_hmm = cvCreate2DHMM( num_states, num_mix, vect_size );
3678
3679            m_vectSize = vect_size;
3680            return true;
3681
3682            //else return false;
3683        }
3684
3685        ////////////////////////////////////////////////////////////////////
3686        // OPENCV 特有の HMM（隠れマルコフモデル）データの入出力
3687        // 今回は XML パーサでなく、ソースコードを改造した自作のパーサで
3688    データを処理
3689        // プログラミングは不効率だが、移植性（MSXML のバージョン問題）
3690    と速度では明白に有利
3691        // 本来 MSXML と DOM 技術を用いより合理的に作成するべきであっ
3692    たが
3693        ////////////////////////////////////////////////////////////////////
3694        // OPENCV 特有の HMM（隠れマルコフモデル）型データの保存（文
3695    字列より）
3696        bool CContEHMM::Save( CString *source )
3697        {
3698            // 入力された文字列と設定するべき HMM 型に対する設定
3699            if (!m_hmm) return false;
3700            if (source == NULL) return false;
```

```
3701
3702                // トポロジー（データ形式）を設定する
3703                source->AppendFormat(_T("%s  %d¥n"),  "<NumSuperStates>",
3704    m_hmm->num_states );
3705                source->AppendFormat(_T("%s "), "<NumStates>");
3706                for( int i = 0; i < m_hmm->num_states; i++ )
3707                    source->AppendFormat(_T("%d                                  "),
3708    m_hmm->u.ehmm[i].num_states );
3709                source->AppendFormat(_T("¥n"));
3710
3711                //各データごとの内部状態の数の読込と全体の内部状態数の保存
3712                source->AppendFormat(_T("%s "), "<NumMixtures>");
3713                for( i = 0; i < m_hmm->num_states; i++ )
3714                {
3715                    CvEHMM* ehmm = &(m_hmm->u.ehmm[i]);
3716                    for( int j = 0; j < ehmm->num_states; j++ )
3717                        source->AppendFormat(_T("%d                                  "),
3718    ehmm->u.state[j].num_mix );
3719                }
3720                source->AppendFormat(_T("¥n") );
3721                source->AppendFormat(_T("%s        %d¥n"),        "<VecSize>",
3722    m_vectSize );   // 異常あり
3723                //errorText.AppendFormat(_T("%s      %d¥n"),      "<VecSize>",
3724    m_vectSize );
3725                    //return false;
3726
3727                    // 全 HMM 情報（隠れマルコフモデル情報）を記述する
3728                    CvEHMM* hmm = m_hmm;
3729                for( i = 0; i < m_hmm->num_states + 1; i++ )
3730                {
3731                    if ( hmm->level == 0 )
3732                        source->AppendFormat(_T("%s¥n"),
3733    "<BeginEmbeddedHMM>");
3734                    else
3735                        source->AppendFormat(_T("%s¥n"),
3736    "<BeginExternalHMM>");
3737                    source->AppendFormat(_T("%s   %d¥n"),   "<NumStates>",
```

```
hmm->num_states );

                if ( hmm->level == 0 )
                {
                    for ( int j = 0; j < hmm->num_states; j++)
                    {
                        CvEHMMState* state = &(hmm->u.state[j]);

                        source->AppendFormat(_T("%s %d¥n"), "<State>",
j);
                        source->AppendFormat(_T("%s              %d¥n"),
"<NumMixes>", state->num_mix);

                        float* mu = state->mu;
                        float* inv_var = state->inv_var;

                        for( int m = 0; m < state->num_mix; m++)
                        {
                                                // 平均（<Mean>）パートの記
述；全ベクトル

        source->AppendFormat(_T("%s %d %s %f¥n"), "<Mixture>", m,
"<Weight>", state->weight[m] );
                                source->AppendFormat(_T("%s¥n"),
"<Mean>");
                                for (int k = 0; k < m_vectSize; k++)

        source->AppendFormat(_T("%f "), *( mu + m * m_vectSize + k ) );

                                                // 加 重 分 散
（<Inverted_Deviation>）パートの記述；全ベクトル
                                source->AppendFormat(_T("¥n"));
                                source->AppendFormat(_T("%s¥n"),
"<Inverted_Deviation>");
                                for (k = 0; k < m_vectSize; k++)
                                    source->AppendFormat(_T("%f          "),
*( inv_var + m * m_vectSize + k ) );
```

```
3775                                source->AppendFormat(_T("¥n"));
3776
3777                                source->AppendFormat(_T("%s          %f¥n"),
3778 "<LogVarVal>", state->log_var_val[m] );
3779
3780                            }
3781                        }
3782                    }
3783
3784                    //推移確率行列（ transition probability matrix）の記述
3785                    source->AppendFormat(_T("%s¥n"), "<TransP>" );
3786                    float* prob = hmm->transP;
3787
3788                    for (int j = 0; j < hmm->num_states; j++)
3789                    {
3790                        for (int k = 0; k < hmm->num_states; k++)
3791                            source->AppendFormat(_T("%f  "),  *(  prob  +  j  *
3792 hmm->num_states + k ) );
3793                        source->AppendFormat(_T("¥n"));
3794                    }
3795
3796                      //終端の記述
3797                    if( hmm->level == 0 )
3798                        source->AppendFormat(_T("%s¥n"),
3799 "<EndEmbeddedHMM>");
3800                    else
3801                        source->AppendFormat(_T("%s¥n"),
3802 "<EndExternalHMM>");
3803
3804                    hmm = &(m_hmm->u.ehmm[i]);
3805                }
3806
3807                return true;
3808            }
3809
3810            // OPENCV 特有の HMM（隠れマルコフモデル）型データの読込（文
3811 字列より）
```

```
3812        bool CContEHMM::Load( CString source )
3813        {
3814            // 必要な変数の設定
3815            int num_states[128];
3816            int num_mix[128];
3817            char temp_char[128];
3818                CString resToken, resToken2;
3819        //文字データの解析
3820                int curPos = 0, curPos2 = 0, i = 0, j = 0;
3821
3822            // 入力された文字列と設定するべき HMM 型に対する設定
3823            if (m_hmm) cvRelease2DHMM( &m_hmm );
3824            if (source.IsEmpty() || source.GetLength() < 5 ) return false;
3825
3826            // トポロジー（要はデータ形式）の読込
3827                resToken = source.Tokenize( "¥n", curPos );
3828            sscanf( resToken, "%s %d ¥n", temp_char, num_states );
3829                resToken = source.Tokenize( "¥n", curPos );
3830                //errorText.AppendFormat(_T("LOAD MESSGAE = %s¥n"),
3831    resToken );
3832                resToken2 = resToken.Tokenize( " ", curPos2 );
3833            for( i = 0; i < num_states[0]; i++ )
3834            {
3835                    if( curPos2 < resToken.GetLength() )
3836                            resToken2    =    resToken.Tokenize(    "    ",
3837    curPos2 );
3838                    sscanf( resToken2, "%d ", num_states + i + 1 );
3839            }
3840
3841            //各データごとの内部状態の数の読込と全体の内部状態数の計算
3842            int total_states = 0;
3843            for( i = 0; i < num_states[0]; i++ )
3844                 total_states += num_states[i+1];
3845
3846            //NUM MIXTURE の処理
3847                curPos2 = 0;
3848                resToken = source.Tokenize( "¥n", curPos );
```

```
3849                    //errorText.AppendFormat(_T("LOAD MESSGAE = %s¥n"),
3850    resToken );
3851                    resToken2 = resToken.Tokenize( " ", curPos2 );
3852                for( i = 0; i < total_states; i++ )
3853                {
3854                            if( curPos2 < resToken.GetLength() )
3855                                    resToken2    =    resToken.Tokenize(    "    ",
3856    curPos2 );
3857                     sscanf( resToken2, "%d ", &num_mix[i] );
3858                }
3859
3860                    //VECT SIZE の処理
3861                    resToken = source.Tokenize( "¥n", curPos );
3862                sscanf( resToken, "%s %d¥n", temp_char, &m_vectSize );
3863
3864                    //HMM 情報クラスのインスタンスを指定されたパラメータに
3865    基づき作成
3866                m_hmm    =    cvCreate2DHMM(    num_states,    num_mix,
3867    m_vectSize );
3868            //!!!          cvCreate2DHMM(  &m_hmm,  num_states,  num_mix,
3869    m_vectSize);
3870                if (!m_hmm ) return false;
3871
3872                    //return false;
3873                //全ての HMM パラメータの読込と設定
3874                CvEHMM* hmm = m_hmm;
3875                    int temp_int;
3876
3877                for( i = 0; i < num_states[0] + 1; i++ )
3878                {
3879                            resToken = source.Tokenize( "¥n", curPos );
3880                            resToken = source.Tokenize( "¥n", curPos );
3881                            //errorText.AppendFormat(_T("LOAD      MESSGAE
3882    = %s¥n"), resToken );
3883                     j = sscanf( resToken, "%s %d¥n", temp_char , &temp_int );
3884
3885                            if( temp_int !=    num_states[i] )
```

```
3886                        {
3887                                errorText.AppendFormat(_T("ERROR      IN
3888  LOADING¥n") );
3889                                return false;
3890                        }
3891
3892                        //後半部分データ読込（ここを利用しない場合前半部分
3893  の表の読込のみを実施）
3894                  if ( i != 0 )
3895                  {
3896                       for ( j = 0; j < num_states[i]; j++)
3897                       {
3898                            CvEHMMState* state = &(hmm->u.state[j]);
3899
3900                                    resToken   =   source.Tokenize(   "¥n",
3901  curPos );
3902                            sscanf(    resToken,    "%s    %d¥n",    temp_char,
3903  &temp_int);
3904                                    if( temp_int != j )
3905                                            return false;
3906
3907                                    resToken   =   source.Tokenize(   "¥n",
3908  curPos );
3909                            sscanf(    resToken,    "%s    %d¥n",    temp_char,
3910  &temp_int );
3911                                    if( temp_int != state->num_mix )
3912                                    {
3913
3914          errorText.AppendFormat(_T("%d¥n"), state->num_mix );
3915
3916          errorText.AppendFormat(_T("ERROR IN LOADING3¥n") );
3917                                            return false;
3918                                    }
3919
3920                            float*   mu   =   (float*)malloc(   state->num_mix   *
3921  m_vectSize * sizeof(float) );
3922                                    float*              inv_var             =
```

```
3923  (float*)malloc( state->num_mix * m_vectSize * sizeof(float) );
3924                                          state->mu = mu;
3925                              state->inv_var = inv_var;
3926
3927                                          //最小繰り返し単位部分のデータ読込
3928                              for( int m = 0; m < state->num_mix; m++)
3929                              {
3930
3931                                                  int temp_int;
3932                                                  resToken                    =
3933  source.Tokenize( "¥n", curPos );
3934                                      sscanf(  resToken,  "%s  %d  %s  %f¥n",
3935  temp_char, &temp_int, temp_char, &(state->weight[m]) );
3936                                                  assert( temp_int == m );
3937
3938                                                  // 平均（<Mean>）パートの読
3939  込；全ベクトル
3940                                                  curPos2 = 0;
3941                                                  resToken                    =
3942  source.Tokenize( "¥n", curPos );
3943                                                  resToken                    =
3944  source.Tokenize( "¥n", curPos );
3945                                      for (int k = 0; k < m_vectSize; k++)
3946                                  {
3947                                                          if(      curPos2      <
3948  resToken.GetLength() )
3949                                                                  resToken2    =
3950  resToken.Tokenize( " ", curPos2 );
3951                                          sscanf(  resToken2,  "%f  ",  (  mu  +  m  *
3952  m_vectSize + k ) );
3953                                  }
3954
3955                                                  //      加      重      分      散
3956   （<Inverted_Deviation>）パートの読込；全ベクトル
3957                                                  curPos2 = 0;
3958                                                  resToken                    =
3959  source.Tokenize( "¥n", curPos );
```

```
3960                                                    resToken                    =
3961    source.Tokenize( "¥n", curPos );
3962                                    for ( k = 0; k < m_vectSize; k++)
3963                                    {
3964                                                            if(     curPos2     <
3965    resToken.GetLength() )
3966                                                                    resToken2   =
3967    resToken.Tokenize( " ", curPos2 );
3968
3969            sscanf( (LPCSTR)resToken2, "%f ", ( inv_var + m * m_vectSize +
3970    k ) );
3971                                    }
3972
3973                                                    resToken                    =
3974    source.Tokenize( "¥n", curPos );
3975                                       sscanf( resToken, "%s %f¥n", temp_char,
3976    &(state->log_var_val[m]) );
3977                                  }
3978                                   }
3979                              }
3980
3981                    // 推移確率行列（transition probability matrix）の読込
3982                            resToken = source.Tokenize( "¥n", curPos );
3983                    float*  prob    =   (float*)malloc(  hmm->num_states    *
3984    hmm->num_states * sizeof(float) );
3985                            hmm->transP = prob;
3986                    for (int j = 0; j < hmm->num_states; j++)
3987                    {
3988                                    curPos2 = 0;
3989                                    resToken = source.Tokenize( "¥n", curPos );
3990                                for (int k = 0; k < hmm->num_states; k++)
3991                            {
3992                                                    if( curPos2 < resToken.GetLength() )
3993                                                            resToken2           =
3994    resToken.Tokenize( " ", curPos2 );
3995                                    sscanf(  resToken2,  "%f  ",  (  prob  +  j  *
3996    hmm->num_states + k ) );
```

```
3997                                    }
3998                                }
3999                                    resToken = source.Tokenize( "¥n", curPos );
4000
4001                                hmm = &(m_hmm->u.ehmm[i]);
4002                            }
4003                            return true;
4004                        }
4005
4006                        BOOL CContEHMM::Release()
4007                        {
4008                            if (m_hmm)
4009                            {
4010                                cvRelease2DHMM( &m_hmm );
4011                                m_hmm = 0;
4012                            }
4013
4014                            return TRUE;
4015                        }
4016    }
4017
4018    ////////////////////////////////////////////////////////////////////////////////
4019    //
4020    //  このプログラムは Intel License Agreement に基づく Intel 社のオープンソ
4021    ースのプログラムである。
4022    //  このプログラムも Intel License Agreement に準じて公開する。
4023    //  なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負
4024    うものとする
4025    //
4026    //  （本ライブラリに必要な DLL ファイル）
4027    //  highgui.dll, cv.dll, cvaux.dll(OPENCV)
4028    //  IPL.dll, IPLa6.dll(IPL)
4029    //
4030    //  （本ファイルの内容)stdafx.h
4031    //  MicroSoft　.NET 用の標準のインクルードファイル
4032    //  （従って、INTEL 社のプログラムとは直接の関連はない）
4033    //
```

```
4034  ////////////////////////////////////////////////////////////////////*/
4035  #pragma once
4036
4037  #using <mscorlib.dll>
4038
4039  ////////////////////////////////////////////////////////////////////////////
4040  //
4041  //  このプログラムは Intel License Agreement に基づく Intel 社のオープンソ
4042  ースのプログラムである。
4043  //  私、飯塚岳郎が卒業研究の目的でいくつかの重要な改変を行った。
4044  //    （１）ATL に基づく COM コンポーネントへの改装
4045  //    （２）ファイルアクセスのデータベース化
4046  //    （３）IPL 導入による BMP<->IPL（OPENCV）間のフォーマット変換機
4047  能の導入
4048  //
4049  //  このプログラムも Intel License Agreement に準じて公開する。
4050  //  なお、このプログラムは INTEL 社とは無関係で、全責任は私飯塚岳郎が負
4051  うものとする
4052  //
4053  //  （本ライブラリに必要な DLL ファイル）
4054  //  highgui.dll, cv.dll, cvaux.dll(OPENCV)
4055  //  IPL.dll, IPLa6.dll(IPL)
4056  //
4057  //  （本ファイルの内容）STDAFX2.H
4058  //  ATL（Active Template Library)用の標準のインクルードファイル
4059  //  もともと顔認識エンジンを COM コンポーネントとして作成したために必
4060  要なファイル
4061  //  （従って、INTEL 社のプログラムとは直接の関連はない）
4062  //
4063  ////////////////////////////////////////////////////////////////////////*/
4064
4065  #pragma once
4066
4067  /* インクルードガード */
4068  #ifndef STRICT
4069  #define STRICT
4070  #endif
```

```
#ifndef FACELIBRARYS
#define FACELIBRARYS

// 下で指定された定義の前に対象プラットフォームを指定しなければならない場合、以下の定義を変更してください。
// 異なるプラットフォームに対応する値に関する最新情報については、MSDN を参照してください。
#ifndef WINVER                  // Windows 95 および Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define WINVER 0x0400           // これを Windows 98 および Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINNT            // Windows NT 4 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINNT 0x0400     // これを Windows 2000 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_WINDOWS          // Windows 98 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINDOWS 0x0410 // これを Windows Me またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#ifndef _WIN32_IE               // IE 4.0 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_IE 0x0400    // これを IE 5.0 またはそれ以降のバージョン向けに適切な値に変更してください。
#endif

#define _ATL_APARTMENT_THREADED
#define _ATL_NO_AUTOMATIC_NAMESPACE

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS      // 一部の CString コンストラクタは明示的です。
```

```
4108
4109  // 一般的で無視しても安全な MFC の警告メッセージの一部の非表示を解除
4110  します。
4111  #define _ATL_ALL_WARNINGS
4112
4113  /* ATL（Active Template Library)用のクラスライブラリの導入 */
4114  #include <atlbase.h>   // ATL の一番基本となるヘッダファイル
4115  #include <atlcom.h>
4116  #include <atlwin.h>
4117  #include <atltypes.h> // ATL::CIMAGE クラスより先にインクルードが必要
4118  #include <atlctl.h>
4119  #include <atlhost.h>
4120  #include <atlcoll.h>    // CATLLIST クラスのため
4121  #include <atlimage.h>// ATL::CIMAGE クラスのため
4122  #include <atlsafe.h>    // CCOMSAFEARRAY クラスのため
4123
4124  using namespace ATL;/* ATL 用の名前空間を導入する */
4125
4126  /* ADO（ActiveX Database Object)：一世代前の MICROSOFT 社の DB イン
4127  ターフェースの導入 */
4128  #import "d:¥Program Files¥Common Files¥System¥ADO¥msado15.dll" ¥
4129      no_namespace rename("EOF", "EndOfFile")
4130  #include <icrsint.h>
4131  #include <stdio.h>
4132
4133  /* OPENCVCV と ATL でクラス名が重複していることへの対策 */
4134  #define ATLIMAGE CImage
4135
4136  /*    OPENCV 関係のヘッダファイル */
4137  #include "cv.h"
4138  #include "vfw.h"
4139  #include "highgui.h"
4140  #define CImage ATLIMAGE
4141
4142  /*    IPL 関係のヘッダファイル OPENCV と共有箇所があるのでその対策も実
4143  施 */
4144  #define OPENCV_INCLUDE 1
```

```
#include "ipl.h"

/*  ASSERT（論理デバッグ）関係の処理 */
#define ASSERT ATLASSERT

#endif
```

（付録 2）Webservice1（WEB サービス本体）

```
'////////////////////////////////////////////////////////////////////////////////
'//
'//　この WEB アプリケーションは飯塚岳郎によって 2004 年 1 月 5 日に作成された。
'//　顔認識アプリケーションのうち WEB サーバー部分を担当するクラスライブラリ。
'//　このプログラムは GPL に準じて公開する。
'//
'//　（本ライブラリに必要な DLL ファイル）
'//　highgui.dll, cv.dll, cvaux.dll(OPENCV)　顔認識ライブラリの DLL
'//　IPL.dll, IPLa6.dll(IPL)
'//
'//　（注意）
'//　現在、このプログラムは顔認識ライブラリを利用することが出来ない。
'//　利用した場合、WEB ページに「構成ファイルが読み込み不能」と報告される
'//
'////////////////////////////////////////////////////////////////////////////*/

'//作業に必要な名前空間（事実上インクルードファイルに相当）
Imports System.Web.Services        'WEB サービスに不可欠
Imports System.IO                  'バイナリデータの取扱いに必要
Imports System.Drawing             '画像データの取扱いに必要
Imports System.Data.SqlClient      'データベースアクセスに必要
'Imports  System.Text              'UNICODE<->SHIFTJIS 変換のため臨時に設置

'//提供される WEB サービス本体
<WebService(Namespace:="http://tempuri.org/")> _
Public Class Service1
    Inherits System.Web.Services.WebService

    'WEB サービス内で共有するべきインスタンス
    Dim connString1 As String = "Data Source=TAKEO01;Initial Catalog=HMMDEMO;uid=test01;pwd=bike0512;"
    Dim errMsg As String = Nothing
```

```
4219        Dim sqlConn As SqlConnection = Nothing
4220        Dim sqlAdpt As SqlDataAdapter = Nothing
4221
4222        'ユースケース記述の「個人の認識」部分のサーバ側担当メソッド
4223        '画像認識サービスを提供する
4224        '実際には画像認識は行わないので指定された適当なキーを送り返すこと
4225    で本来の動作と代用
4226        <WebMethod()> Public Function FaceRecognize(ByVal image As Byte())
4227    As String
4228            '画像の受信
4229            Dim mStream As System.IO.MemoryStream = New
4230    System.IO.MemoryStream(image)
4231            Dim imageData As Bitmap = New Bitmap(mStream)
4232
4233            '作業に必要なデータの確保
4234            Dim person() As Long = {1, 2, 3, 4, 5, 6}
4235            Dim likelihood() As Single = {0.54, 0.57, 0.59}
4236            Dim information() As String
4237            Dim result As String
4238            Dim i As Integer
4239
4240            '結果返還用文字列の作成
4241            information = dataLoader(person)
4242            For i = 0 To 2
4243                result += information(i)
4244                'result += likelihood(i).ToString
4245                result += "¥"
4246            Next
4247            FaceRecognize = result
4248
4249        End Function
4250
4251        '//ユースケース記述の「個人情報の検索」部分のサーバ側担当メソッド
4252        '//指定されたキーを元に個人情報を提供するサービスを提供する
4253        '//実際にはパターン学習は行わないのでその命令は記述されない
4254        <WebMethod()> Public Function GetPersonInfo(ByVal key As Integer)
4255    As String
```

```
4256
4257            '作業に必要なデータの確保
4258            Dim information() As String
4259            Dim result As String
4260            Dim i As Integer
4261
4262            '結果返還用文字列の作成
4263            GetPersonInfo = PersonLoader(key)
4264
4265        End Function
4266
4267        '//ユースケース記述の「個人情報の修正」部分のサーバ側担当メソッド
4268        '//指定されたキー、文字列を元にデータベース内の個人情報を修正する
4269        <WebMethod()> Public Function PutPersonInfo(ByVal key As Integer,
4270    ByVal name As String, ByVal belong As String, ByVal relation As String,
4271    ByVal address As String, ByVal tel As String) As String
4272
4273            '作業に必要なデータの確保
4274            Dim data As Human = New Human()
4275            Dim result As String
4276            Dim i As Integer
4277
4278            '結果返還用文字列の作成
4279            data.setValue(name, belong, relation, address, tel)
4280            PutPersonInfo = PersonWriter(key, data)
4281
4282        End Function
4283
4284
4285        '//ユースケース記述の「携帯電話上からの新規個人の登録」部分のサーバ
4286    側担当メソッド
4287        '//指定されたキー、文字列を元にデータベース内に新たな個人を登録する
4288        'その際、個人情報テーブルと顔認識テーブルの両方を更新
4289        <WebMethod()> Public Function SetNewPerson(ByVal name As String,
4290    ByVal belong As String, ByVal relation As String, ByVal address As String,
4291    ByVal tel As String) As String
4292
```

```
4293            '作業に必要なデータの確保
4294            Dim data As Human = New Human()
4295            Dim result As String
4296            Dim key As Long
4297
4298
4299            '結果返還用文字列の作成
4300            data.setValue(name, belong, relation, address, tel)
4301            key = SetPersonOnPerson()
4302            SetNewPerson = PersonSetter(key, data)
4303
4304        End Function
4305
4306
4307        '//ユースケース記述の「携帯電話上からの新規個人の登録」部分に個人登
4308    録テーブル操作を提供
4309        '//SetNewPerson で起動され、個人情報テーブルに新規レコードを付加
4310        Private Function PersonSetter(ByVal key As Integer, ByRef data As
4311    Human) As String
4312            '接続用 SQL インスタンスの作成
4313            Dim i As Integer
4314            Dim word As String
4315            PersonSetter = "OK"
4316
4317            'データベース接続
4318            Dim dSet As DataSet = DBLOADER("SELECT * FROM
4319    INFORMATION", "INFORMATION")
4320
4321            '氏名を検証する(完全に同姓同名なら排除)
4322            Dim length As Long =
4323    dSet.Tables("INFORMATION").Columns(0).MaxLength()
4324            For i = 0 To length
4325                word = dSet.Tables("INFORMATION").Rows(0).Item(" 氏 名
4326    ").ToString()
4327                If word = data.name Then
4328                    PersonSetter = "PROBLEM"
4329                    Exit Function
```

```
4330                End If
4331            Next
4332
4333            '新しい DataRow オブジェクトを作成する
4334            Dim dRow As DataRow =
4335    dSet.Tables("INFORMATION").NewRow()
4336            dRow.Item("氏名") = data.name
4337            dRow.Item("PERSONID") = key
4338            dRow.Item("所属") = data.belong
4339            dRow.Item("関係") = data.relation
4340            dRow.Item("住所") = data.address
4341            dRow.Item("電話番号") = data.tel
4342            dSet.Tables("INFORMATION").Rows.Add(dRow)
4343
4344            ' データベースを更新する
4345            DBUPLOADER(dSet, "INFORMATION")
4346
4347        End Function
4348
4349
4350        '//ユースケース記述の「携帯電話上からの新規個人の登録」部分に顔認識
4351    テーブル操作を提供
4352        '//SetNewPerson で起動され、顔認識テーブルに新規レコードを付加し、
4353    その主キーを入手
4354        '//この方式による SQL プログラミングは極めて悪趣味（無駄な計算量大）
4355        '//移植の都合上配慮したが、SQLSERVER で利用するなら更新時のキー関
4356    係の SQL コマンドを修正せよ
4357        Private Function SetPersonOnPerson() As Long
4358            '接続用 SQL インスタンスの作成
4359            Dim i As Integer
4360            SetPersonOnPerson = -1
4361
4362            'データベース接続
4363            Dim dSet As DataSet = DBLOADER("SELECT * FROM PERSON",
4364    "PERSON")
4365
4366            '新しい DataRow オブジェクトを作成する
```

```
4367            Dim dRow As DataRow = dSet.Tables("PERSON").NewRow()
4368            dRow.Item("NAME") = CType("NULL", String)
4369            dSet.Tables("PERSON").Rows.Add(dRow)
4370
4371            ' データベースを更新する
4372            DBUPLOADER(dSet, "PERSON")
4373
4374            'データベース接続
4375            dSet = DBLOADER("SELECT * FROM PERSON ORDER BY
4376    PERSONID DESC", "PERSON")
4377            SetPersonOnPerson                                    =
4378    dSet.Tables("PERSON").Rows(0).Item("PERSONID")
4379        End Function
4380
4381
4382
4383        '//個人情報テーブルのデータベース更新関数
4384        '//更新内容は文字列ではなく専用の Human クラスと呼ばれるクラスで入
4385    力する
4386        Private Function PersonWriter(ByRef key As Long, ByRef data As
4387    Human) As String
4388            '接続文字列の作成
4389            Dim comString As String = "SELECT * FROM INFORMATION "
4390            Dim infoString As String
4391            infoString += "WHERE INFORMATIONID = "
4392            infoString += key.ToString
4393            comString += infoString '"WHERE PERSONID = 1 or PERSONID
4394    = 4"
4395
4396            'データベース接続
4397            Dim dSet As DataSet
4398            dSet = DBLOADER(comString, "INFORMATION")
4399
4400            '新しい DataRow オブジェクトを作成する
4401            dSet.Tables("INFORMATION").Rows(0).Item("氏名") = data.name
4402            dSet.Tables("INFORMATION").Rows(0).Item("所属") = data.belong
4403            dSet.Tables("INFORMATION").Rows(0).Item("　関　係　")　=
```

```
4404    data.relation
4405            dSet.Tables("INFORMATION").Rows(0).Item("　住　所　") =
4406    data.address
4407            dSet.Tables("INFORMATION").Rows(0).Item("電話番号") =
4408    data.tel
4409
4410            ' データベースを更新する
4411            PersonWriter = DBUPLOADER(dSet, "INFORMATION")
4412        End Function
4413
4414
4415
4416        '//個人情報テーブルのデータベース探索関数
4417        '//（ユースケース記述「個人情報の検索」用）
4418        Private Function PersonLoader(ByRef key As Long) As String
4419            '接続文字列の作成
4420            Dim comString As String = "SELECT * FROM INFORMATION "
4421            Dim infoString As String
4422            infoString += "WHERE INFORMATIONID = "
4423            infoString += key.ToString
4424            comString += infoString '"WHERE PERSONID = 1 or PERSONID
4425    = 4"
4426
4427            'データベース接続
4428            Dim sqlAdpt As SqlDataAdapter = New SqlDataAdapter()
4429            Dim dSet As DataSet
4430            dSet = DBLOADER(comString, "INFORMATION")
4431
4432            ' DataSet からデータを入手する
4433            PersonLoader  =  dSet.Tables("INFORMATION").Rows(0).Item("氏
4434    名").ToString()
4435            PersonLoader += ","
4436            PersonLoader += dSet.Tables("INFORMATION").Rows(0).Item("所
4437    属").ToString()
4438            PersonLoader += ","
4439            PersonLoader += dSet.Tables("INFORMATION").Rows(0).Item("関
4440    係").ToString()
```

```
4441            PersonLoader += ","
4442            PersonLoader += dSet.Tables("INFORMATION").Rows(0).Item("住
4443    所").ToString()
4444            PersonLoader += ","
4445            PersonLoader += dSet.Tables("INFORMATION").Rows(key).Item("
4446    電話番号").ToString()
4447        End Function
4448
4449
4450        '//個人情報テーブルのデータベース探索関数
4451        '//（ユースケース記述「個人の認識」用）
4452        Private Function dataLoader(ByRef person() As Long) As String()
4453            '結果返還用文字列配列の作成
4454            Dim result(3) As String
4455            Dim i As Integer
4456            For i = 0 To 2
4457                result(i) = Nothing
4458            Next
4459
4460            '接続文字列の作成
4461            Dim comString As String = "SELECT PERSONID, 氏名, 所属, 関
4462    係 FROM INFORMATION "
4463            Dim infoString As String = "WHERE "
4464            For i = 0 To 2
4465                infoString += "PERSONID = "
4466                infoString += person(i).ToString
4467                If (i <> 2) Then
4468                    infoString += " or "
4469                End If
4470            Next
4471            comString += infoString
4472
4473            'データベース接続
4474            Dim sqlAdpt As SqlDataAdapter = New SqlDataAdapter()
4475            Dim dSet As DataSet
4476            dSet = DBLOADER(comString, "INFORMATION")
4477
```

```
4478            ' DataSet からデータを入手する
4479            Dim resultString As String
4480            For i = 0 To 2
4481                Dim rowSTring As String
4482                rowSTring  =  dSet.Tables("INFORMATION").Rows(i).Item("氏
4483 名").ToString()
4484                rowSTring += ","
4485                rowSTring  +=  dSet.Tables("INFORMATION").Rows(i).Item("
4486 所属").ToString()
4487                rowSTring += ","
4488                rowSTring                                                 +=
4489 dSet.Tables("INFORMATION").Rows(i).Item("PERSONID").ToString()
4490                rowSTring += ","
4491                result(i) = rowSTring
4492            Next
4493
4494            'System.Text.UTF8Encoding.Convert(
4495            ' 結果を報告する
4496            dataLoader = result
4497        End Function
4498
4499
4500        '//データベース接続業務を提供
4501        Private Function DBLOADER(ByRef comString As String, ByRef table
4502 As String) As DataSet
4503            '接続用 SQL インスタンスの作成
4504            sqlConn = New SqlConnection(connString1)
4505            sqlAdpt = New SqlDataAdapter()
4506            Dim dSet As DataSet = New DataSet()
4507
4508            'データベース準備
4509            sqlAdpt.SelectCommand = New SqlCommand()
4510            With sqlAdpt.SelectCommand
4511                .CommandText = comString
4512                .CommandType = CommandType.Text
4513                .Connection = sqlConn
4514            End With
```

```
4515
4516            'データベース接続
4517            Try
4518                sqlConn.Open()
4519                sqlAdpt.Fill(dSet, table)
4520            Catch ex As Exception
4521                sqlConn.Close()
4522                errMsg = ex.Message
4523                DBLOADER = Nothing
4524                Exit Function
4525            End Try
4526
4527            sqlConn.Close()
4528            DBLOADER = dSet
4529        End Function
4530
4531
4532        '//データベース更新業務を提供
4533        Private Function DBUPLOADER(ByRef dSet As DataSet, ByRef table
4534    As String) As String
4535
4536            'データベース更新文の作成
4537            Dim cBuild As SqlCommandBuilder = New
4538    SqlCommandBuilder(sqlAdpt)
4539
4540            'データベース接続と更新
4541            Try
4542                sqlAdpt.Update(dSet, table)      ' INSERT ステートメントを
4543    サブミットする
4544                dSet.AcceptChanges()              ' DB への変更が完了したこ
4545    とを通知する
4546                sqlConn.Close()
4547            Catch err As Exception
4548                sqlConn.Close()
4549                DBUPLOADER = err.Message
4550                Exit Function
4551            End Try
```

```
4552
4553            DBUPLOADER = "OK"
4554        End Function
4555
4556        '//データベース接続業務を代行(SQL 文の単純実行用)
4557        Private Function DBREQUIRE(ByRef comString As String) As String
4558
4559            '接続用 SQL インスタンスの作成
4560            sqlConn = New SqlConnection(connString1)
4561            Dim sqlReader As SqlDataReader
4562            Dim sqlCmd As SqlCommand = New SqlCommand()
4563            With sqlCmd
4564                .CommandText = comString
4565                .CommandType = CommandType.Text
4566                .Connection = sqlConn
4567            End With
4568
4569            'データベース接続
4570            Try
4571                sqlConn.Open()
4572                sqlReader = sqlCmd.ExecuteReader()
4573                DBREQUIRE = sqlReader.GetString(0)
4574            Catch ex As Exception
4575                sqlConn.Close()
4576                DBREQUIRE = ex.Message
4577                Exit Function
4578            End Try
4579
4580            sqlConn.Close()
4581        End Function
4582
4583    End Class
4584
4585    '//この WEB アプリケーションで個人情報を格納するためのクラス
4586    Public Class Human
4587        Public name As String
4588        Public belong As String
```

```
        Public relation As String
        Public address As String
        Public tel As String
        Public key As String

        '//データを設定するためのメソッド
        Public Sub setValue(ByVal names As String, ByVal belongs As String,
ByVal relations As String, ByVal addresss As String, ByVal tels As String)
            name = names
            belong = belongs
            relation = relations
            address = addresss
            tel = tels
        End Sub
End Class
```

（付録 3）KSOAP シリーズ（携帯電話上のクライアント）

```
/*////////////////////////////////////////////////////////////////////
// KSOAP1.java
// 日付 : 2004/01/18 0:16
// 作者 : 飯塚　岳郎
// 目的 : ユースケース「個人の認識」の携帯電話側プログラムの実装
// 参照 : 「i アプリプログラミング」などを参照（詳細は本文参考資料を参照）
////////////////////////////////////////////////////////////////////*/

//携帯電話関係のクラスライブラリを導入
import java.io.*;
import java.util.*;
import com.nttdocomo.io.*;              //携帯電話の基本的なクラスライブラ
リ
import com.nttdocomo.ui.*;
import com.nttdocomo.device.Camera;          //携帯電話のカメラ関係
import com.nttdocomo.opt.device.Camera2;
import javax.microedition.io.*;              //携帯電話のデータ操作を担当

//KSOAP 関係のクラスライブラリを導入
import org.ksoap.*;
import org.ksoap.transport.*;
import org.ksoap.marshal.*;
import org.kobjects.user.*;
import org.kobjects.serialization.*;

public class KSOAP1 extends IApplication
{
        //必要な定数(WEB サービス接続先)
        String w_url = "http://localhost/Webservice1/Service1.asmx";
        String ww_name = "";
        HttpConnection wi_connect;

        //事実上のメイン関数
        public void start()
        {
                //自分自身をインスタンス化し、そのメソッドを実行
```

```
4663                    KSOAP1 myapp = new KSOAP1();
4664
4665                    //ユースケース「個人の認識」の携帯電話側動作を実行
4666                    InputStream w_in = myapp.imageProc();                   // 撮 影
4667    と画像保存
4668                    if( w_in == null )
4669                            terminate();
4670                    byte[] data = myapp.LoadImage( w_in );                  // 画 像
4671    の配列読込
4672                    String ret = myapp.FaceRecognize( w_url, data );//顔認識の
4673    依頼
4674                    System.out.println("実行結果:" + ret);          //顔認識結果の
4675    表示
4676
4677                    myapp.showResult( ret );
4678                    //terminate();
4679            }
4680
4681            //WEB サービス FaceRecognize（「個人の認識」）に相当へ接続
4682            private String FaceRecognize( String w_url, byte[] data )
4683            {
4684                    Object ret = null;
4685
4686                    try
4687                    {
4688                            //SOAP 接続用設定設定
4689                            ElementType type = new ElementType();
4690                            String soapAction = "http://tempuri.org/";
4691                                    //サーバプログラム側で指定する名前空間らし
4692    い。
4693                            HttpTransport http = new HttpTransport( w_url,
4694    soapAction );   // Transport は KSOAP で用意されたクラス
4695                            ClassMap classMap= new ClassMap( Soap.VER11 );
4696                    //ClassMap も KSOAP で用意されたクラス
4697                            http.setClassMap(classMap);
4698                            http.debug = false;
4699
```

```
4700                    //SOAP 実行関数の指定と実行
4701                    SoapObject method = new SoapObject( soapAction,
4702 "FaceRecognize" );
4703                    method.addProperty("image",          soapAction,
4704 (Object)data);
4705                    ret = (Object)http.call(method);
4706                }
4707                catch(Exception e)
4708                {
4709                    System.err.println(e.toString());
4710                }
4711
4712                String convert = ret.toString();
4713
4714                return ret.toString();
4715        }
4716
4717        //画像撮影・処理メソッド
4718        public InputStream imageProc()
4719        {
4720                InputStream image_in = null;
4721                Image w_image;
4722                Camera w_camera;
4723
4724                //カメラの設定
4725                w_camera = Camera2.getCamera(0);                    // カ メ
4726 ラの複数存在可能性あり
4727                w_camera.setImageSize( 192, 144 );
4728
4729                //撮影の実行
4730                try
4731                {
4732                    w_camera.takePicture();
4733                }
4734                catch( Exception ie )   //Interrupted
4735                {
4736                    System.out.println( "撮影に問題発生" );
```

```
4737                    }
4738
4739                    //画像の読込
4740                    try {
4741                            //本来の形式とは異なるが画像の読込
4742                            MediaImage                  w_media                 =
4743    MediaManager.getImage("resource:///test041.jpg");
4744                            image_in                                            =
4745    Connector.openInputStream( "resource:///test041.jpg" );;
4746                            System.out.println( "撮影完了" );
4747                            w_media.use();
4748
4749                            //画像の表示と結果の出力
4750                            w_image = w_media.getImage();
4751                    }
4752                    catch( Exception e )
4753                    {
4754                            System.out.println("画像処理に何らかのトラブル");
4755                            System.out.println(  " メ ッ セ ー ジ   =  "  +
4756    e.getMessage() );
4757                    }
4758
4759                    return image_in;
4760            }
4761
4762            //画像の配列形式への取りまとめ
4763            public byte[] LoadImage( InputStream w_in )
4764            {
4765                    byte[] w_data = new byte[1];
4766
4767                    //ストリームを用いた読込作業の遂行
4768                    try
4769                    {
4770                            w_data = new byte[5000];
4771                            w_in.read( w_data );
4772                    }
4773                    catch( IOException ce )
```

```
4774                {
4775                        System.out.println("読込に問題発生");
4776                }
4777
4778                return w_data;
4779        }
4780
4781        //結果の表示、管理用メソッド
4782        public void showResult(String data)
4783        {
4784                //最終結果の表示準備
4785                mypanel w_panel = new mypanel();
4786                w_panel.setSoftLabel( w_panel.SOFT_KEY_1, "終了" );
4787                w_panel.setSoftLabel( w_panel.SOFT_KEY_2, "個人情報" );
4788
4789                //受信データの設定と表示
4790                w_panel.setValue(data);
4791                w_panel.showValue();
4792
4793                //最終結果の表示
4794                w_panel.w_listbox.requestFocus();
4795                Display.setCurrent( w_panel );
4796
4797        }
4798
4799        ////////////////////////////////////////////////////////////////////////
4800        // 受信データを保持するクラス
4801        // 受信データの解析も行う
4802        ////////////////////////////////////////////////////////////////////////
4803        class PersonData
4804        {
4805                //利用したい情報
4806                int key;                //主キー
4807                String Name;            //氏名
4808                String Belong;          //所属
4809
4810                //受信データの解析
```

```
4811                    PersonData( String data )
4812                    {
4813                            try{
4814                                    StringTokenizer st2 = new
4815    StringTokenizer( data,",");
4816                                    Name = st2.nextToken();
4817                                    Belong = st2.nextToken();
4818                                    //key = 1;
4819                                    key = Integer.parseInt(st2.nextToken());
4820                            }
4821                            catch(Exception e){
4822
4823                            }
4824                    }
4825            }
4826
4827            ////////////////////////////////////////////////////////////////
4828            // 表示画面（GUI）を記述するクラス
4829            // このプログラムで必要な次の画面への遷移なども実装する
4830            ////////////////////////////////////////////////////////////////
4831            class mypanel extends Panel
4832                    implements ComponentListener, SoftKeyListener,
4833    KeyListener
4834            {
4835                    //変数（構成部品）の設定
4836                    private Label w_label;
4837                    private ListBox w_listbox;
4838                    private TextBox w_textbox;
4839                    PersonData[] personList;
4840                    int key_value;
4841
4842                    //描画動作(非表示状態での描画動作を実行)
4843                    public mypanel()
4844                    {
4845                            //リストボックス
4846                            w_label = new Label("候補者リスト¥n");
4847            //ラベル
```

```
4848
4849            w_label.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
4850                        add( w_label );
4851                        w_listbox                   =                   new
4852    ListBox( ListBox.SINGLE_SELECT );//リストボックス
4853
4854            w_listbox.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
4855                        add( w_listbox );
4856
4857                        //イベントリスナーのインスタンス登録
4858                        setSoftKeyListener( this );
4859                        setKeyListener( this );
4860                        setComponentListener( this );
4861                }
4862
4863                //結果の表示、管理用メソッド
4864                public void setValue(String data)
4865                {
4866                        //受信データの解析
4867                        StringTokenizer          st          =          new
4868    StringTokenizer(data,"¥¥");
4869                        personList = new PersonData[3];
4870                        for( int i = 0; i < 3 && st.hasMoreTokens(); i++ )
4871                        {
4872                                personList[i]               =               new
4873    PersonData(st.nextToken());
4874                        }
4875                }
4876
4877                //結果の表示用メソッド
4878                public void showValue()
4879                {
4880                        //受信データの表示
4881                        for( int i = 0; i < 3; i++ )
4882                        {
4883                                w_listbox.append("  氏  名  :  "  +
4884    personList[i].Name);
```

```
4885                                    w_listbox.append("　所　属　：　"　+
4886    personList[i].Belong);
4887                            }
4888                    }
4889
4890                    //ユースケース記述「個人情報の検索」を担当する GUI の起動
4891    操作
4892                    void showPersonInfo( int key )
4893                    {
4894                            String[] args = new String[3];
4895                            args[0] = "http://localhost/KSOAP2.jam";
4896                            args[1] = "getKey";
4897                            args[2] = Integer.toString(key);
4898
4899            IApplication.getCurrentApp().launch( IApplication.LAUNCH_IAPP
4900    LI, args );
4901                    }
4902
4903                    //ソフトキー動作の記述（この 2 関数は SoftKeyListener イン
4904    ターフェース設定上必須）
4905                    //アプリケーションではプログラムの動作終了、次のプログラ
4906    ムへの移動を担当するボタン
4907                    public void softKeyReleased( int param )
4908                    {
4909                            //キー操作への応対
4910                            if( param == this.SOFT_KEY_1 )
4911                            {
4912                                    terminate();
4913                            }
4914                            if( param == this.SOFT_KEY_2 )
4915                            {
4916                                    showPersonInfo( personList[key_value].key );
4917                            }
4918                    }
4919
4920                    public void softKeyPressed( int param )
4921                    {
```

```
4922                }
4923
4924                //コンポーネント動作の記述（ComponentListener インターフ
4925    ェース設定上必須）
4926                public void componentAction( Component w_component, int
4927    w_type, int w_param )
4928                {
4929                        //該当者を示すリストボックス操作への応対
4930                        if( ( w_component == w_listbox ) && ( w_type ==
4931    SELECTION_CHANGED ) )
4932                        {
4933                                if( w_param == 0 || w_param == 1 )
4934                                {
4935                                        key_value = 0;
4936                                }
4937                                else if( w_param == 2 || w_param == 3 )
4938                                {
4939                                        key_value = 1;
4940                                }
4941                                else if( w_param == 4 || w_param == 5 )
4942                                {
4943                                        key_value = 2;
4944                                }
4945                        }
4946                }
4947
4948                public void keyPressed( Panel w_panel, int w_param )
4949                {
4950                        //キー操作への応対
4951                        if( w_param == Display.KEY_1 )
4952                        {
4953                                showPersonInfo( personList[0].key );
4954                        }
4955                        if( w_param == Display.KEY_2 )
4956                        {
4957                                showPersonInfo( personList[1].key );
4958                        }
```

```
4959                            if( w_param == Display.KEY_3 )
4960                            {
4961                                    showPersonInfo( personList[2].key );
4962                            }
4963                    }
4964
4965                    public void keyReleased( Panel w_panel, int w_param )
4966                    {
4967                    }
4968
4969            }
4970    }
4971
4972    /*//////////////////////////////////////////////////////////////////////
4973    // KSOAP2.java
4974    // 日付 : 2004/01/19 05:36
4975    // 作者 : 飯塚　岳郎
4976    // 目的 : ユースケース「個人情報の検索、修正」の携帯電話側プログラムの実
4977    装
4978    // 参照 : 「i アプリプログラミング」などを参照（詳細は本文参考資料を参照）
4979    //////////////////////////////////////////////////////////////////////*/
4980
4981    //携帯電話関係のクラスライブラリを導入
4982    import java.io.*;
4983    import java.util.*;
4984    import com.nttdocomo.io.*;              //携帯電話の基本的なクラスライブラ
4985    リ
4986    import com.nttdocomo.ui.*;
4987
4988    //KSOAP 関係のクラスライブラリを導入
4989    import org.ksoap.*;
4990    import org.ksoap.transport.*;
4991    import org.ksoap.marshal.*;
4992    import org.kobjects.user.*;
4993    import org.kobjects.serialization.*;
4994
4995    public class KSOAP2 extends IApplication
```

```
{
	//必要な定数(WEB サービス接続先)
	String w_url = "http://localhost/Webservice1/Service1.asmx";
	String ww_name = "";

	//事実上のメイン関数
	public void start()
	{
		//自分自身をインスタンス化し、そのメソッドを実行
		KSOAP2 myapp = new KSOAP2();

		//パラメータとして渡されたキーを元に実際の業務を遂行
		Long data = new
Long(Long.parseLong(IApplication.getCurrentApp().getParameter("getKey
")));
		String ret = myapp.GetPersonInfo( w_url, data );	//WEB
サービスの呼び出し
		myapp.showResult( data.longValue(), ret );		// 結 果
の表示
		//terminate();
	}

	//WEB サービス GetPersonInfo（「個人情報の検索」相当）へ接続
	private String GetPersonInfo( String w_url, Long data )
	{
		Object ret = null;

		try
		{
			//SOAP 接続用設定設定
			ElementType type = new ElementType();
			String soapAction = "http://tempuri.org/";
	//サーバプログラム側で指定する名前空間らしい。
			HttpTransport http = new HttpTransport( w_url,
soapAction );	// Transport は KSOAP で用意されたクラス
			ClassMap classMap= new ClassMap( Soap.VER11 );
		//ClassMap も KSOAP で用意されたクラス
```

```
5033                        http.setClassMap(classMap);
5034                        http.debug = false;
5035
5036                        //SOAP 実行関数の指定と実行
5037                        SoapObject method = new SoapObject( soapAction,
5038 "GetPersonInfo" );
5039                        method.addProperty("key",                soapAction,
5040 (Object)data);
5041                        ret = (Object)http.call(method);
5042                }
5043                catch(Exception e)
5044                {
5045                        System.err.println(e.toString());
5046                }
5047
5048                return ret.toString();
5049        }
5050
5051        //WEB サービス SetPersonInfo（「個人情報の修正」相当）へ接続
5052        private String PutPersonInfo( String w_url, PersonData data )
5053        {
5054                Object ret = null;
5055                Long key = new Long(data.key);
5056
5057                try
5058                {
5059                        //SOAP 接続用設定設定
5060                        ElementType type = new ElementType();
5061                        String soapAction = "http://tempuri.org/";
5062        //サーバプログラム側で指定する名前空間らしい。
5063                        HttpTransport http = new HttpTransport( w_url,
5064 soapAction );   // Transport は KSOAP で用意されたクラス
5065                        ClassMap classMap= new ClassMap( Soap.VER11 );
5066                //ClassMap も KSOAP で用意されたクラス
5067                        http.setClassMap(classMap);
5068                        http.debug = false;
5069
```

```
5070                    //SOAP 実行関数の指定と実行
5071                    //最初の空白値は、soapAction と共通する名前空間関係
5072 の引数らしいが、やっぱり用意してない。
5073                    SoapObject method = new SoapObject( soapAction,
5074 "PutPersonInfo" );
5075                    method.addProperty("key", soapAction, (Object)key);
5076                    method.addProperty("name",          soapAction,
5077 (Object)data.Name);
5078                    method.addProperty("belong",        soapAction,
5079 (Object)data.Belong);
5080                    method.addProperty("relation",      soapAction,
5081 (Object)data.Relation);
5082                    method.addProperty("tel",           soapAction,
5083 (Object)data.TEL);
5084                    method.addProperty("address",       soapAction,
5085 (Object)data.Address);
5086                    ret = (Object)http.call(method);
5087            }
5088            catch(Exception e)
5089            {
5090                    System.err.println(e.toString());
5091            }
5092
5093            return ret.toString();
5094    }
5095
5096    //結果の表示、管理用メソッド
5097    public void showResult(long key, String data)
5098    {
5099            //最終結果の表示準備
5100            mypanel w_panel = new mypanel();
5101            w_panel.setSoftLabel( w_panel.SOFT_KEY_1, "終了" );
5102            w_panel.setSoftLabel( w_panel.SOFT_KEY_2, "送信" );
5103
5104            //受信データの設定
5105            w_panel.setValue(key, data);
5106            w_panel.showValue();
```

```
5107
5108                    //受信データの表示
5109                    w_panel.w_listbox.requestFocus();
5110                    Display.setCurrent( w_panel );
5111            }
5112
5113            ////////////////////////////////////////////////////////////////////
5114            // WEB サーバと送受信するデータを保持するクラス
5115            // 受信データの解析も行う
5116            ////////////////////////////////////////////////////////////////////
5117            class PersonData
5118            {
5119                    //利用したい情報
5120                    long key = 0;                   //主キー
5121                    String Name= null;              //氏名
5122                    String Belong= null;            //所属
5123                    String Relation= null;          //間柄
5124                    String TEL= null;               //電話番号
5125                    String Address= null;           //住所
5126
5127                    //受信データの解析
5128                    PersonData( long key_input, String data )
5129                    {
5130                            key = key_input;
5131
5132                            StringTokenizer st2 = new
5133    StringTokenizer( data,"," );
5134                            Name = st2.nextToken();
5135                            Belong = st2.nextToken();
5136                            Relation = st2.nextToken();
5137                            Address = st2.nextToken();
5138                            TEL = st2.nextToken();
5139                    }
5140
5141                    //データの代入
5142                    void setData( int key, String value )
5143                    {
```

```
                        switch( key ){
                        case 0: Name = value; break;
                        case 1: Belong = value; break;
                        case 2: Relation = value; break;
                        case 3: TEL = value; break;
                        case 4: Address = value; break;
                        }
                }
        }

        ////////////////////////////////////////////////////////////////////
        // 終了画面（元へもどる）の GUI を記述するプログラム
        ////////////////////////////////////////////////////////////////////
        class mypanel_finish extends Panel implements SoftKeyListener
        {
                //変数（構成部品）の設定
                private Label w_label;

                public mypanel_finish(String data)
                {
                        w_label = new Label(data);

        w_label.setFont( Font.getFont( Font.SIZE_LARGE ) );
                        add( w_label );

                        //イベントリスナーのインスタンス登録
                        setSoftKeyListener( this );
                }

                //ソフトキー動作の記述（この 2 関数は SoftKeyListener インターフェース設定上必須）
                //アプリケーションではプログラムの動作終了、次のプログラムへの移動を担当するボタン
                public void softKeyReleased( int param )
                {
                        //キー操作への応対
                        if( param == this.SOFT_KEY_2 )
```

```
5181                                    terminate();
5182                            if( param == this.SOFT_KEY_1 )
5183                            {
5184                                    String[] args = new String[1];
5185                                    args[0] = "http://localhost/KSOAP.jam";
5186
5187            IApplication.getCurrentApp().launch(  IApplication.LAUNCH_IAPP
5188    LI, args );
5189                            }
5190                    }
5191
5192                    public void softKeyPressed( int param )
5193                    {
5194                    }
5195            }
5196
5197            ////////////////////////////////////////////////////////////////////
5198            // 表示画面（GUI）を記述するクラス
5199            // キーなどの操作に対応するメソッドもここに記述
5200            // このプログラムで必要な次の画面への遷移なども実装する
5201            ////////////////////////////////////////////////////////////////////
5202            class mypanel extends Panel
5203                    implements ComponentListener, SoftKeyListener
5204            {
5205                    //変数（構成部品）の設定
5206                    private Label w_label, w_label2;
5207                    private Button w_button, w_button2;
5208                    private ListBox w_listbox;
5209                    private TextBox w_textbox;
5210                    PersonData list;
5211
5212                    //描画動作(非表示状態での描画動作を実行)
5213                    public mypanel()
5214                    {
5215                            //該当者を表示するリストボックス
5216                            w_label = new Label("個人情報画面¥n");
5217            //ラベル
```

```
5218
5219            w_label.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5220                        add( w_label );
5221                        w_listbox                   =                   new
5222    ListBox( ListBox.SINGLE_SELECT );//リストボックス
5223
5224            w_listbox.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5225                        add( w_listbox );
5226
5227                        //修正する内容を入力するテキストボックス
5228                        w_label2 = new Label("情報入力¥n");          // ラ ベ
5229    ル
5230
5231            w_label2.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5232                        add( w_label2 );
5233                        w_textbox    =    new    TextBox(    "",    32,    1,
5234    TextBox.DISPLAY_ANY );
5235
5236            w_textbox.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5237                        add( w_textbox );
5238
5239                        //作業の選択肢となるボタン
5240                        w_button = new Button("情報修正");
5241            //ボタン 1
5242                        add( w_button );
5243                        w_button2 = new Button("新規登録");          // ボ タ
5244    ン 2
5245                        add( w_button2 );
5246
5247                        //イベントリスナーのインスタンス登録
5248                        setSoftKeyListener( this );     //ソフトキー関係
5249                        setComponentListener( this );//GUI 操作関係
5250
5251                }
5252
5253                //結果の管理用メソッド
5254                public void setValue(long key, String data)
```

```
5255                {
5256                        //受信データの解析
5257                        list = new PersonData(key, data);
5258                }
5259
5260                //結果の表示用メソッド
5261                public void showValue()
5262                {
5263                        //受信データの表示
5264                        w_listbox.append("氏名：" + list.Name);
5265                        w_listbox.append("所属：" + list.Belong);
5266                        w_listbox.append("関係：" + list.Relation);
5267                        w_listbox.append("TEL：" + list.TEL);
5268                        w_listbox.append("住所：" + list.Address);
5269                }
5270
5271                //ソフトキー動作の記述（この 2 関数は SoftKeyListener イン
5272  ターフェース設定上必須）
5273                //アプリケーションではプログラムの動作終了、次のプログラ
5274  ムへの移動を担当するボタン
5275                public void softKeyReleased( int param )
5276                {
5277                        //キー操作への応対
5278                        if( param == this.SOFT_KEY_1 )
5279                        {
5280                                mypanel_finish w_panel = new
5281  mypanel_finish("元へ戻る");
5282
5283          w_panel.setSoftLabel( w_panel.SOFT_KEY_1, "戻る" );
5284
5285          w_panel.setSoftLabel( w_panel.SOFT_KEY_2, "終了" );
5286                                Display.setCurrent( w_panel );
5287
5288                        }
5289                        if( param == this.SOFT_KEY_2 )
5290                        {
5291                                KSOAP2 myapp = new KSOAP2();
```

```
5292                                String value = myapp.PutPersonInfo( w_url,
5293    list );
5294                                if( value.equals("OK") )
5295                                {
5296                                        //最終結果の表示準備
5297                                        mypanel_finish  w_panel  =  new
5298    mypanel_finish("変更完了");
5299
5300            w_panel.setSoftLabel( w_panel.SOFT_KEY_1, "戻る" );
5301
5302            w_panel.setSoftLabel( w_panel.SOFT_KEY_2, "終了" );
5303                                        Display.setCurrent( w_panel );
5304                                }
5305                        }
5306                }
5307
5308                public void softKeyPressed( int param )
5309                {
5310                }
5311
5312                //コンポーネント動作の記述（ComponentListener インターフ
5313    ェース設定上必須）
5314                public void componentAction( Component w_component, int
5315    w_type, int w_param )
5316                {
5317
5318                        //ボタン操作への応対
5319                        //ユースケース記述「新規個人の登録」に該当するプロ
5320    グラムを起動
5321                        if( ( w_component == w_button2 ) && ( w_type ==
5322    BUTTON_PRESSED ) )
5323                        {
5324                                String[] args = new String[1];
5325                                args[0] = "http://localhost/KSOAP3.jam";
5326
5327            IApplication.getCurrentApp().launch( IApplication.LAUNCH_IAPP
5328    LI, args );
```

```
5329                        }
5330
5331                        //リストボックス操作への応対
5332                        //GUI 操作を合理化した（選択後、テキストボックスへ
5333    即移動）
5334                        if( ( w_component == w_listbox ) && ( w_type ==
5335    SELECTION_CHANGED ) )
5336                        {
5337                                w_textbox.requestFocus();
5338                        }
5339
5340                        //テキストボックス操作への応対
5341                        //GUI 操作を合理化した（入力終了後、選択した選択肢
5342    のすぐ下へ移動）
5343                        if( ( w_component == w_textbox ) && ( w_type ==
5344    TEXT_CHANGED   ) )
5345                        {
5346                                System.out.println(w_textbox.getText());
5347
5348            System.out.println(w_listbox.getSelectedIndex());
5349                                list.setData(     w_listbox.getSelectedIndex(),
5350    w_textbox.getText() );
5351                                w_listbox.removeAll();
5352                                showValue();
5353                                w_listbox.requestFocus();
5354                                Display.setCurrent( this );
5355                        }
5356                }
5357        }
5358    }
5359
5360    /*//////////////////////////////////////////////////////////////////
5361    // KSOAP3.java
5362    // 日付 : 2004/01/19 05:36
5363    // 作者 : 飯塚　岳郎
5364    // 目的 : ユースケース「新規個人の登録」の携帯電話側プログラムの実装
5365    // 参照 : 「i アプリプログラミング」などを参照（詳細は本文参考資料を参照）
```

```
5366  ////////////////////////////////////////////////////////////////*/
5367
5368  //携帯電話関係のクラスライブラリを導入
5369  import java.io.*;
5370  import java.util.*;
5371  import com.nttdocomo.io.*;
5372  import com.nttdocomo.ui.*;
5373
5374  //KSOAP 関係のクラスライブラリを導入
5375  import org.ksoap.*;
5376  import org.ksoap.transport.*;
5377  import org.ksoap.marshal.*;
5378  import org.kobjects.user.*;
5379  import org.kobjects.serialization.*;
5380
5381  public class KSOAP3 extends IApplication
5382  {
5383          //必要な定数(WEB サービス接続先)
5384          String w_url = "http://localhost/Webservice1/Service1.asmx";
5385          String ww_name = "";
5386
5387          //事実上のメイン関数
5388          public void start()
5389          {
5390                  //最終結果の表示準備
5391                  mypanel w_panel = new mypanel();
5392                  w_panel.setSoftLabel( w_panel.SOFT_KEY_1, "終了" );
5393                  w_panel.setSoftLabel( w_panel.SOFT_KEY_2, "送信" );
5394                  Display.setCurrent( w_panel );
5395
5396                  //terminate();
5397          }
5398
5399          //WEB サービス SetNewPerson（「新規個人の登録」相当）へ接続
5400          private String SetNewPerson( String w_url, PersonData data )
5401          {
5402                  Object ret = null;
```

```
5403                    Long key = new Long(data.key);
5404
5405                    try
5406                    {
5407                            //SOAP 接続用設定設定
5408                            ElementType type = new ElementType();
5409                            String soapAction = "http://tempuri.org/";
5410            //サーバプログラム側で指定する名前空間らしい。
5411                            HttpTransport http = new HttpTransport( w_url,
5412    soapAction );   // Transport は KSOAP で用意されたクラス
5413                            ClassMap classMap= new ClassMap( Soap.VER11 );
5414                    //ClassMap も KSOAP で用意されたクラス
5415                            http.setClassMap(classMap);
5416                            http.debug = false;
5417
5418                            //SOAP 実行関数の指定と実行
5419                            SoapObject method = new SoapObject( soapAction,
5420    "SetNewPerson" );
5421                            method.addProperty("name",              soapAction,
5422    (String)data.Name);
5423                            method.addProperty("belong",            soapAction,
5424    (String)data.Belong);
5425                            method.addProperty("relation",          soapAction,
5426    (String)data.Relation);
5427                            method.addProperty("tel",               soapAction,
5428    (String)data.TEL);
5429                            method.addProperty("address",           soapAction,
5430    (String)data.Address);
5431                            ret = (Object)http.call(method);
5432                    }
5433                    catch(Exception e)
5434                    {
5435                            System.err.println(e.toString());
5436                    }
5437
5438                    return ret.toString();
5439            }
```

```
5440
5441            ////////////////////////////////////////////////////////////////////
5442            // WEB サーバと送受信するデータを保持するクラス
5443            // 受信データの解析も行う
5444            ////////////////////////////////////////////////////////////////////
5445            class PersonData
5446            {
5447                    //利用したい情報
5448                    long key = 0;                   //主キー
5449                    String Name= null;              //氏名
5450                    String Belong= null;            //所属
5451                    String Relation= null;          //間柄
5452                    String TEL= null;               //電話番号
5453                    String Address= null;           //住所
5454
5455                    //単純なコンストラクタ
5456                    PersonData()
5457                    {
5458                            //利用したい情報
5459                            long key = 0;                   //主キー
5460                            String Name= "NULL";                    //氏名
5461                            String Belong= "NULL";                  //所属
5462                            String Relation= "NULL";        //間柄
5463                            String TEL= "NULL";             //電話番号
5464                            String Address= "NULL";                 //住所
5465                    }
5466
5467                    //受信データの解析
5468                    PersonData( long key_input, String data )
5469                    {
5470                            key = key_input;
5471
5472                            StringTokenizer         st2         =           new
5473    StringTokenizer( data,"," );
5474                            Name = st2.nextToken();
5475                            Belong = st2.nextToken();
5476                            Relation = st2.nextToken();
```

```
5477                    Address = st2.nextToken();
5478                    TEL = st2.nextToken();
5479            }
5480
5481
5482            //データの代入
5483            void setData( int key, String value )
5484            {
5485                    switch( key ){
5486                    case 0: Name = value; break;
5487                    case 1: Belong = value; break;
5488                    case 2: Relation = value; break;
5489                    case 3: TEL = value; break;
5490                    case 4: Address = value; break;
5491                    }
5492            }
5493    }
5494
5495    ////////////////////////////////////////////////////////////////////
5496    // 終了画面（元へもどる）の GUI を記述するプログラム
5497    ////////////////////////////////////////////////////////////////////
5498    class mypanel_finish extends Panel implements SoftKeyListener
5499    {
5500            //変数（構成部品）の設定
5501            private Label w_label;
5502
5503            public mypanel_finish(String data)
5504            {
5505                    w_label = new Label(data);
5506
5507    w_label.setFont( Font.getFont( Font.SIZE_LARGE ) );
5508                    add( w_label );
5509
5510                    //イベントリスナーのインスタンス登録
5511                    setSoftKeyListener( this );
5512            }
5513
```

```
5514                    //ソフトキー動作の記述（この 2 関数は SoftKeyListener イン
5515    ターフェース設定上必須）
5516                    //アプリケーションではプログラムの動作終了、次のプログラ
5517    ムへの移動を担当するボタン
5518                    public void softKeyReleased( int param )
5519                    {
5520                            //キー操作への応対
5521                            if( param == this.SOFT_KEY_2 )
5522                                    terminate();
5523                            if( param == this.SOFT_KEY_1 )
5524                            {
5525                                    String[] args = new String[1];
5526                                    args[0] = "http://localhost/KSOAP.jam";
5527
5528            IApplication.getCurrentApp().launch(  IApplication.LAUNCH_IAPP
5529    LI, args );
5530                            }
5531                    }
5532
5533                    public void softKeyPressed( int param )
5534                    {
5535                    }
5536            }
5537
5538            ////////////////////////////////////////////////////////////////
5539            // 表示画面（GUI）を記述するクラス
5540            // キーなどの操作に対応するメソッドもここに記述
5541            // このプログラムで必要な次の画面への遷移なども実装する
5542            ////////////////////////////////////////////////////////////////
5543            class mypanel extends Panel
5544                    implements ComponentListener, SoftKeyListener
5545            {
5546                    //変数（構成部品）の設定
5547                    private Label w_label, w_label2;
5548                    private ListBox w_listbox;
5549                    private TextBox w_textbox;
5550                    PersonData list;
```

```
5551
5552                    //描画動作(非表示状態での描画動作を実行)
5553                    public mypanel()
5554                    {
5555                            //送信する内容を表示するリストボックス
5556                            w_label = new Label("個人情報画面¥n");
5557            //ラベル
5558
5559            w_label.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5560                            add( w_label );
5561                            w_listbox                       =                       new
5562    ListBox( ListBox.SINGLE_SELECT );//リストボックス
5563
5564            w_listbox.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5565                            list = new PersonData();
5566                            showValue();
5567                            add( w_listbox );
5568
5569                            //修正する内容を入力するテキストボックス
5570                            w_label2 = new Label("情報入力¥n");             // ラ ベ
5571    ル
5572
5573            w_label2.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5574                            add( w_label2 );
5575                            w_textbox       =       new       TextBox(       "",       32,       1,
5576    TextBox.DISPLAY_ANY );
5577
5578            w_textbox.setFont( Font.getFont( Font.SIZE_MEDIUM ) );
5579                            add( w_textbox );
5580
5581                            //イベントリスナーのインスタンス登録
5582                            setSoftKeyListener( this );     //ソフトキー関係
5583                            setComponentListener( this );//GUI 操作関係
5584
5585                    }
5586
5587                    //結果の管理用メソッド
```

```
5588                public void setValue(long key, String data)
5589                {
5590                        //受信データの解析
5591                        list = new PersonData(key, data);
5592                }
5593
5594                //結果の表示用メソッド
5595                public void showValue()
5596                {
5597                        //受信データの表示
5598                        w_listbox.append("氏名：" + list.Name);
5599                        w_listbox.append("所属：" + list.Belong);
5600                        w_listbox.append("関係：" + list.Relation);
5601                        w_listbox.append("TEL：" + list.TEL);
5602                        w_listbox.append("住所：" + list.Address);
5603                }
5604
5605                //ソフトキー動作の記述（この 2 関数は SoftKeyListener イン
5606  ターフェース設定上必須）
5607                //アプリケーションではプログラムの動作終了、次のプログラ
5608  ムへの移動を担当するボタン
5609                public void softKeyReleased( int param )
5610                {
5611                        //キー操作への応対
5612                        if( param == this.SOFT_KEY_2 )
5613                        {
5614                                //KSOAP3 のメソッドを利用するため、あえて
5615  インスタンス化
5616                                //メモリ消費上問題なので適切な対策を講ずる
5617  べき
5618                                KSOAP3 myapp = new KSOAP3();
5619
5620
5621                                //WEB サービスを経由して新規個人を登録
5622
5623                                String ret = myapp.SetNewPerson( w_url,
5624  list );
```

```
5625                                    System.out.println("実行結果:" + ret);
5626                                    if( ret.equals("PROBLEM") )
5627                                            ret = "登録拒否";
5628                                    else
5629                                            ret = "登録完了";
5630
5631
5632                                    //結果の報告
5633                                    mypanel_finish      w_panel      =      new
5634    mypanel_finish(ret);
5635
5636            w_panel.setSoftLabel( w_panel.SOFT_KEY_1, "元へ" );
5637
5638            w_panel.setSoftLabel( w_panel.SOFT_KEY_2, "終了" );
5639                                    Display.setCurrent( w_panel );
5640                            }
5641                            if( param == this.SOFT_KEY_1 )
5642                            {
5643                                    //「元へ戻る」画面の起動
5644                                    mypanel_finish      w_panel      =      new
5645    mypanel_finish("元へ戻りますか？");
5646
5647            w_panel.setSoftLabel( w_panel.SOFT_KEY_1, "元へ" );
5648
5649            w_panel.setSoftLabel( w_panel.SOFT_KEY_2, "終了" );
5650                                    Display.setCurrent( w_panel );
5651                            }
5652                    }
5653
5654                    public void softKeyPressed( int param )
5655                    {
5656                    }
5657
5658                    //コンポーネント動作の記述（ComponentListener インターフ
5659    ェース設定上必須）
5660                    public void componentAction( Component w_component, int
5661    w_type, int w_param )
```

```
5662                {
5663                        //リストボックス操作への応対
5664                        //GUI 操作を合理化した（選択後、テキストボックスへ
5665    即移動）
5666                        if( ( w_component == w_listbox ) && ( w_type ==
5667    SELECTION_CHANGED ) )
5668                        {
5669                                w_textbox.requestFocus();
5670                        }
5671
5672                        //テキストボックス操作への応対
5673                        //GUI 操作を合理化した（入力終了後、選択した選択肢
5674    のすぐ下へ移動）
5675                        if( ( w_component == w_textbox ) && ( w_type ==
5676    TEXT_CHANGED   ) )
5677                        {
5678                                int select_key = w_listbox.getSelectedIndex();
5679                                list.setData(                   select_key,
5680    w_textbox.getText() );
5681                                w_listbox.removeAll();
5682                                showValue();
5683
5684                                w_listbox.requestFocus();
5685                                if( select_key < 4 )
5686                                        w_listbox.select( select_key + 1 );
5687                                else
5688                                        w_listbox.select( 0 );
5689
5690                                Display.setCurrent( this );
5691                        }
5692                }
5693        }
5694    }
5695
5696    /* kSOAP
5697     *
5698     * The contents of this file are subject to the Enhydra Public License
```

```
5699  * Version 1.1 (the "License"); you may not use this file except in
5700  * compliance with the License. You may obtain a copy of the License
5701  * on the Enhydra web site ( http://www.enhydra.org/ ).
5702  *
5703  * Software distributed under the License is distributed on an "AS IS"
5704  * basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.
5705 See
5706  * the License for the specific terms governing rights and limitations
5707  * under the License.
5708  *
5709  * The Initial Developer of kSOAP is Stefan Haustein. Copyright (C)
5710  * 2000, 2001 Stefan Haustein, D-46045 Oberhausen (Rhld.),
5711  * Germany. All Rights Reserved.
5712  *
5713  * Contributor(s): John D. Beatty, Dave Dash, F. Hunter, Renaud Tognelli,
5714  *                 Thomas Strang, Alexander Krebs, Sean McDaniel
5715  *
5716  * (補足条項）
5717  *  飯塚　岳郎によって NTTDOCOMO 対応仕様に改造された。
5718  *
5719  * (プログラムの名称）HttpTransport.java
5720  * (プログラムの内容）KSOAP クライアントで通信機能を管轄
5721  * (改造者）　　飯塚　岳郎
5722  * (最終改造日付）　　2004/01/19 05:36
5723  * */
5724
5725 package org.ksoap.transport;
5726
5727 /* NTT の携帯電話で通信に必要なクラスライブラリ  */
5728 import java.io.*;
5729 import javax.microedition.io.*;
5730 import com.nttdocomo.net.*;
5731 import com.nttdocomo.io.*;
5732
5733 /* KSOAP の機能を果たすのに必要なクラス-ライブラリ  */
5734 import org.kxml.*;
5735 import org.kxml.io.*;
```

```
5736  import org.kxml.parser.*;
5737  import org.ksoap.*;
5738
5739  public class HttpTransport {
5740
5741      String url;
5742      String soapAction = "¥"¥"";
5743
5744      SoapEnvelope requestEnvelope = new SoapEnvelope ();
5745      SoapEnvelope responseEnvelope = new SoapEnvelope ();
5746
5747      HttpConnection connection;
5748      OutputStream os;
5749      InputStream is;
5750      //InputStream--
5751      Reader reader;
5752
5753      /** state info */
5754      private boolean connected = false;
5755
5756      /** Set to true if debugging */
5757      public boolean debug;
5758
5759      /** String dump of request for debugging. */
5760      public String requestDump;
5761
5762      /** String dump of response for debugging */
5763      public String responseDump;
5764
5765
5766      /** default constructor */
5767      public HttpTransport () {
5768      }
5769
5770
5771      /**
5772       * Creates instance of HttpTransport with set url and SoapAction
```

```
5773          *
5774          * @param url the destination to POST SOAP data
5775          * @param soapAction the desired SOAP action (for HTTP headers)
5776          */
5777
5778         public HttpTransport (String url, String soapAction) {
5779             this.url = url;
5780             this.soapAction = soapAction;
5781         }
5782
5783
5784         /**
5785          * Set the target url.
5786          *
5787          * @param url the target url.
5788          */
5789
5790         public void setUrl (String url) {
5791             this.url = url;
5792         }
5793
5794
5795         /**
5796          * set the desired soapAction header field
5797          *
5798          * @param soapAction the desired soapAction
5799          */
5800
5801         public void setSoapAction (String soapAction) {
5802             this.soapAction = soapAction;
5803         }
5804
5805
5806         /**
5807          * set the desired ClassMap for the SOAP Envelopes
5808          *
5809          * @param classMap the desired ClassMap
```

```
5810         */
5811
5812        public void setClassMap (ClassMap classMap) {
5813            requestEnvelope.setClassMap (classMap);
5814            responseEnvelope.setClassMap (classMap);
5815        }
5816
5817
5818        /** Sends the requestEnvelope and fills the
5819            responseEnvelope
5820             @exception InterruptedIOException if transport was closed async.
5821            @exception IOException if an error occurs
5822        */
5823
5824        public void call () throws IOException {
5825
5826            //作業に必要な変数、インスタンスの宣言
5827            ByteArrayOutputStream bos = new ByteArrayOutputStream ();
5828            XmlWriter xw = new XmlWriter (new OutputStreamWriter (bos));
5829            String ww_name = "";
5830            requestEnvelope.write (xw);
5831            xw.flush ();
5832            bos.write ('¥r');
5833            bos.write ('¥n');
5834            byte [] requestData = bos.toByteArray ();
5835            bos = null;
5836            xw = null;
5837
5838            requestDump = debug ? new String (requestData) : null;
5839            responseDump = null;
5840
5841             try {
5842                //HTML ヘッダの作成
5843                connection = (HttpConnection)
5844                            Connector.open                (url                +
5845    URLEncoder.encode(ww_name), Connector.READ_WRITE, true);
5846                 connection.setRequestMethod (HttpConnection.POST);
```

```
5847                connection.setRequestProperty("Content-Type",          "text/xml;
5848    charset=Shift-JIS¥r¥nSOAPAction: ¥"" + soapAction + "¥"¥r¥n");
5849                 os = connection.openOutputStream ();
5850
5851                //SOAP 要求メッセージの送信
5852                os.write (requestData, 0, requestData.length);    // BASE64 の場
5853    合、送信メッセージサイズの制限にひっかかる。
5854
5855                //os.flush ();    // removed in order to avoid chunked encoding
5856                 os.close ();
5857                 requestData = null;
5858                connection.connect();
5859                 connected = true;
5860
5861                 // SOAP 返答メッセージの受信
5862                receive();
5863            }
5864            catch( InterruptedIOException iie )
5865            {
5866                    System.out.println("何らかのトラブル:通信系統");
5867            }
5868            catch( ConnectionException ce )
5869            {
5870                    System.out.println("何らかのトラブル:相手側のサーバー");
5871                     receive();        // エラー報告の受信
5872            }
5873             finally
5874            {
5875                if (!connected) throw new InterruptedIOException ();
5876                reset ();
5877             }
5878
5879        }
5880
5881        // データの受信とパーサーへの取り込み
5882        public void receive () throws IOException {
5883
```

```
5884            ByteArrayOutputStream bos = new ByteArrayOutputStream ();
5885             is = connection.openInputStream();
5886             bos = new ByteArrayOutputStream ();
5887
5888             // データの受信
5889             byte [] buf = new byte [256];
5890             while (true) {
5891                  int rd = is.read (buf, 0, 256);
5892                  if (rd == -1) break;
5893                  bos.write (buf, 0, rd);
5894             }
5895
5896             // 配列形式への受信データの変換
5897             buf = bos.toByteArray ();
5898             is.close ();
5899             is = new ByteArrayInputStream (buf);
5900             connection.close();
5901
5902             // 受信データの分析
5903             is = new ByteArrayInputStream (buf);
5904             reader = new BufferedReader( new InputStreamReader (is) );
5905             XmlParser xp = new XmlParser (reader);
5906             responseEnvelope.parse (xp);
5907        }
5908
5909        /**
5910         * Executes a SOAP Method and returns a response
5911         *
5912         * @param method the remote soap method to be executed
5913         * @return the result of the soap method
5914         * @exception IOException if an error occurs
5915         */
5916
5917        public Object call (SoapObject method) throws IOException {
5918
5919            //WEB サーバへの要求
5920            requestEnvelope.setBody (method);                    //受信データの
```

```
5921 獲得;
5922         soapAction = method.getNamespace() + method.getName();   //WEB
5923 サーバへの要求を作成;
5924         call ();                                                  //送信プログラムの呼
5925 び出し
5926
5927         //WEB サーバの返答の処理
5928         if (responseEnvelope.getBody () instanceof SoapFault)
5929             throw((SoapFault)responseEnvelope.getBody ());
5930
5931         //System.out.println("分析以外は完了");
5932         return responseEnvelope.getResult();
5933     }
5934
5935
5936     public void call (XmlIO request, XmlIO result) throws IOException {
5937         requestEnvelope.setBody (request);
5938         responseEnvelope.setBody (result);
5939
5940         if (responseEnvelope.getBody () instanceof SoapFault)
5941             throw((SoapFault)responseEnvelope.getBody ());
5942     }
5943
5944
5945     /**
5946      * Closes the connection and associated streams.   This method
5947      * does not need to be explictly called since the uderlying
5948      * connections and streams are only opened and valid inside of
5949      * the call method.   Close can be called ansynchronously,
5950      * from another thread to potentially release another thread
5951      * that is hung up doing network io inside of call.   Caution
5952      * should be taken, however when using this as a psedu timeout
5953      * mechanism.   it is a valid and suggested approach for the
5954      * motorola handsets.   oh, and it works in the emulator...
5955      */
5956     public void reset () {
5957         connected = false;
```

```
5958            if (reader != null) {
5959                try { reader.close (); }
5960                catch (Throwable e) { }
5961                reader = null;
5962            }
5963
5964            if (is != null) {
5965                try { is.close (); }
5966                catch (Throwable e) { }
5967                is = null;
5968            }
5969
5970            if (connection != null) {
5971                try { connection.close (); }
5972                catch (Throwable e) { }
5973                connection = null;
5974            }
5975        }
5976
5977    }
5978
```